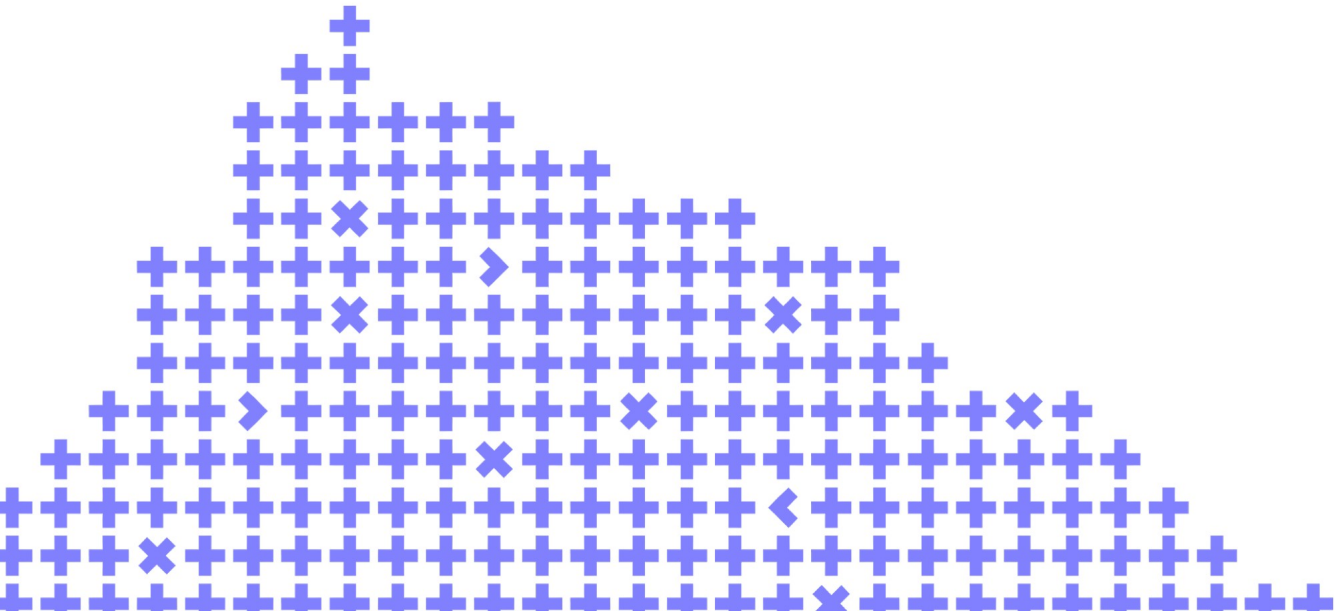
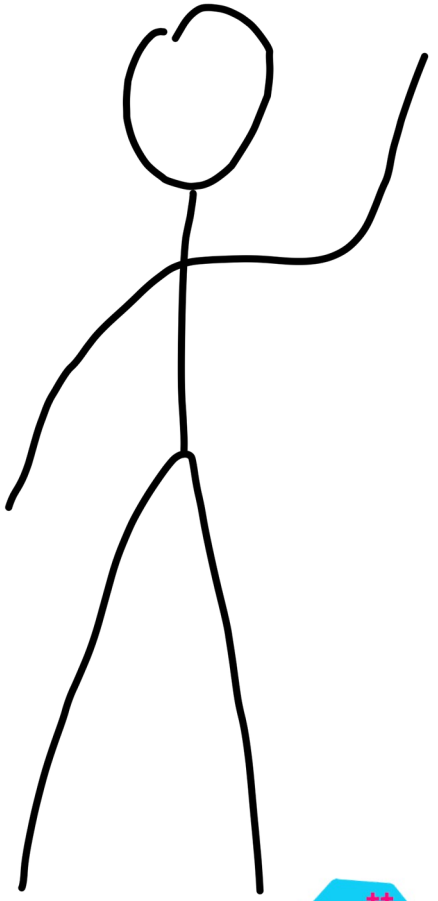
How to choose a queue *properly*

Mons Anderson @ Tarantool



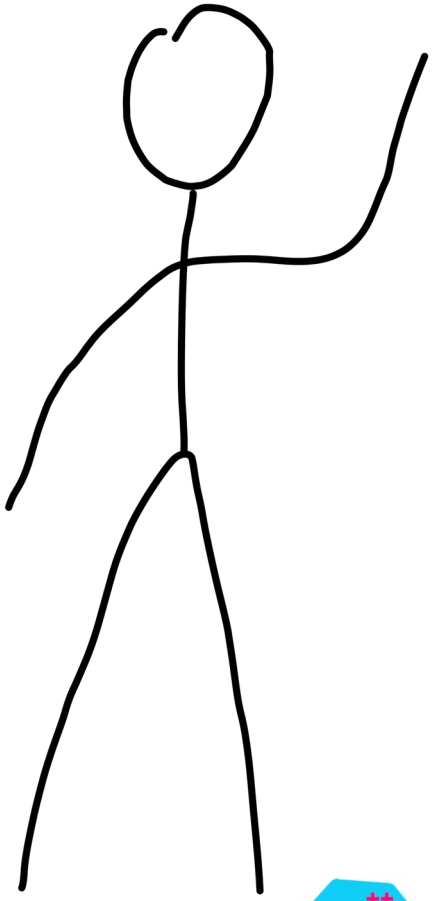
Who am I?

- VK Cloud Storage Architect
- Tarantool Architect



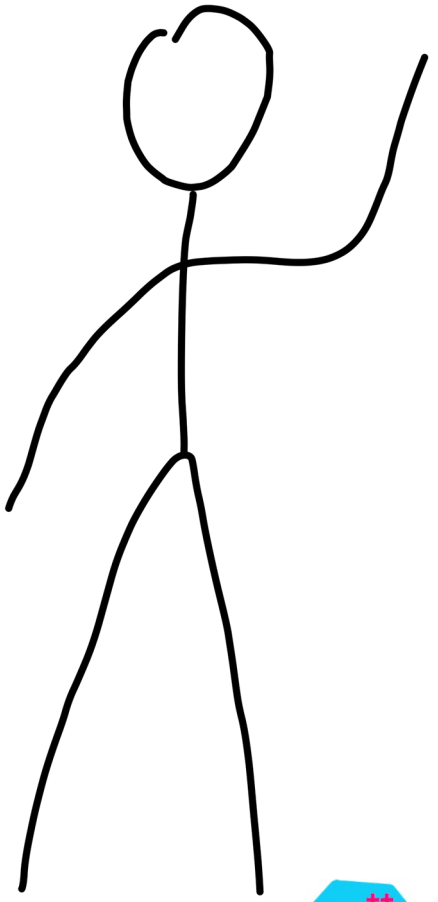
Who am I?

- VK Cloud Storage Architect
- Tarantool Architect & PM



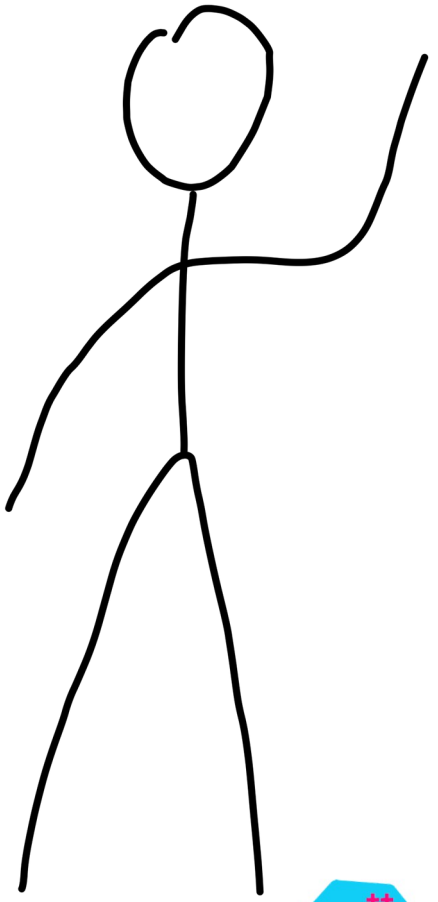
Who am I?

- VK Cloud Storage Architect
- Tarantool Architect & PM & Evangelist



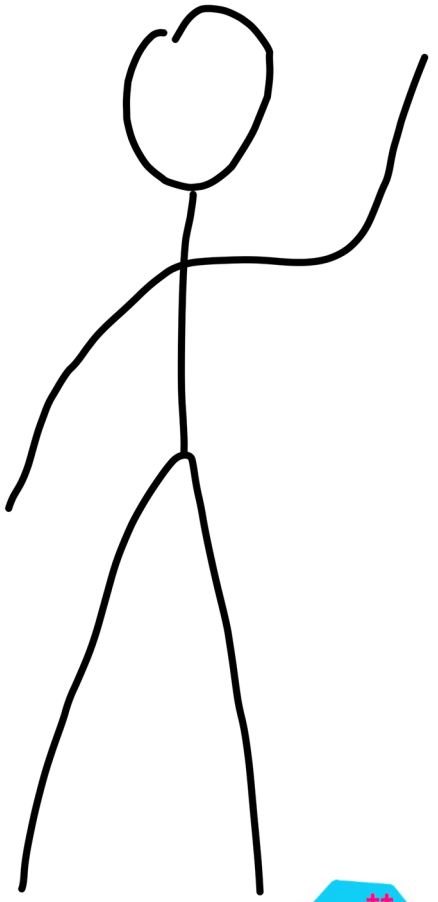
Who am I?

- VK Cloud Storage Architect
- Tarantool Architect & PM & Evangelist
- I use queues since 2008



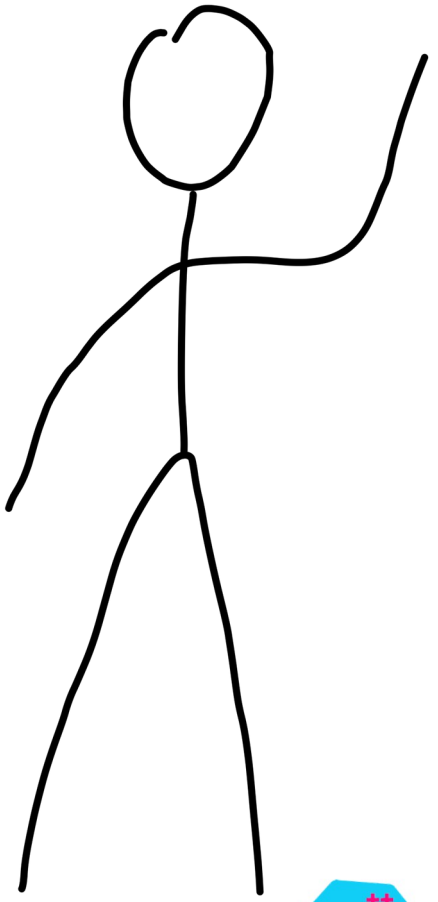
Who am I?

- VK Cloud Storage Architect
- Tarantool Architect & PM & Evangelist
- I use queues since 2008
- I like to create queues



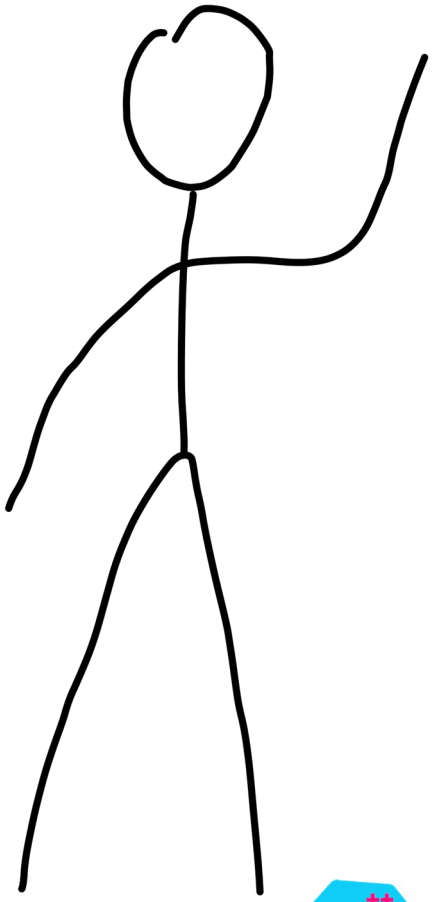
Who am I?

- VK Cloud Storage Architect
- Tarantool Architect & PM & Evangelist
- I use queues since 2008
- I like to create queues (esp. with **Tarantool**)



Who am I?

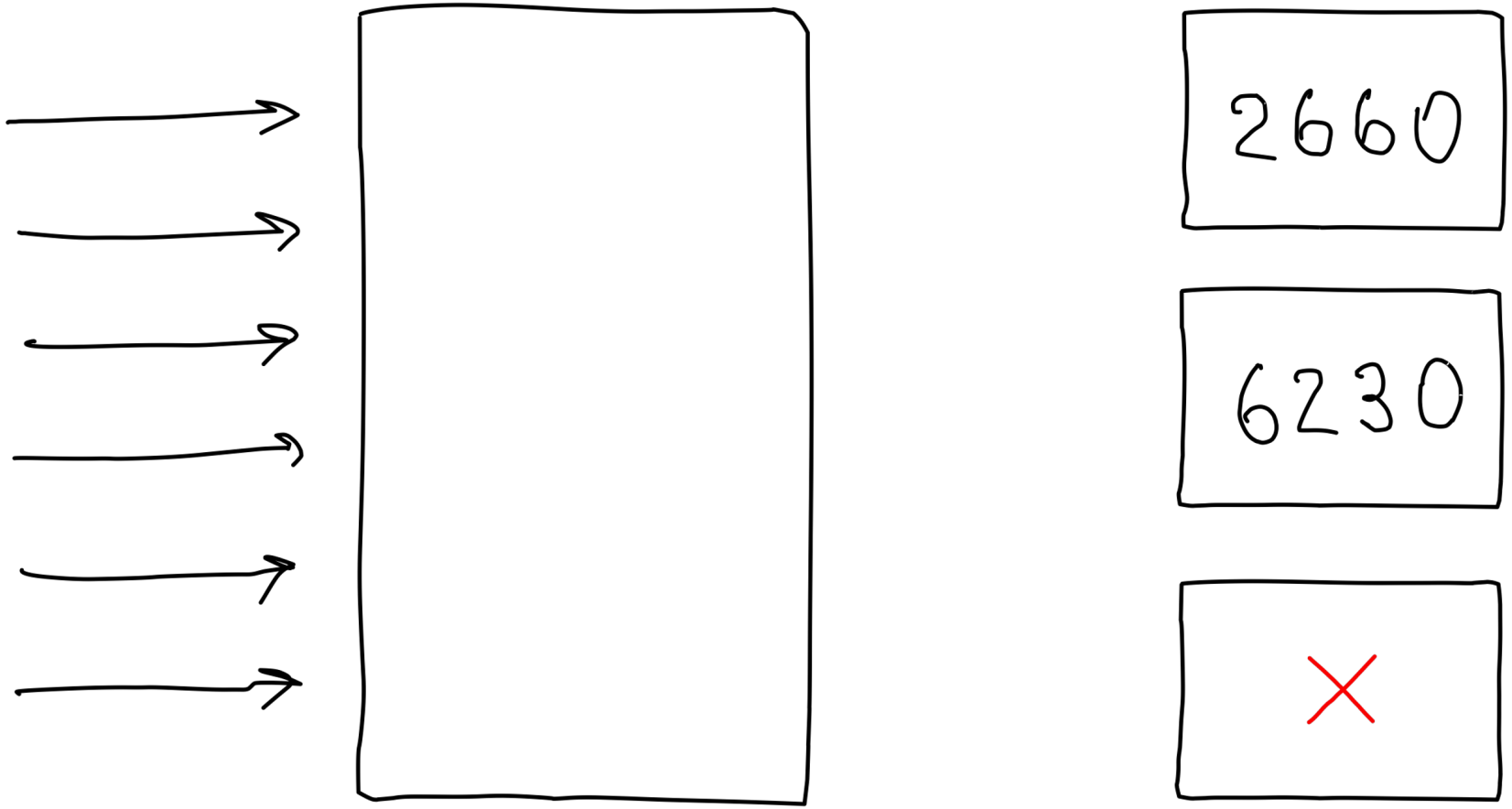
- VK Cloud Storage Architect
- Tarantool Architect & PM & Evangelist
- I use queues since 2008
- I like to create queues (esp. with Tarantool)
- I like to talk about queues



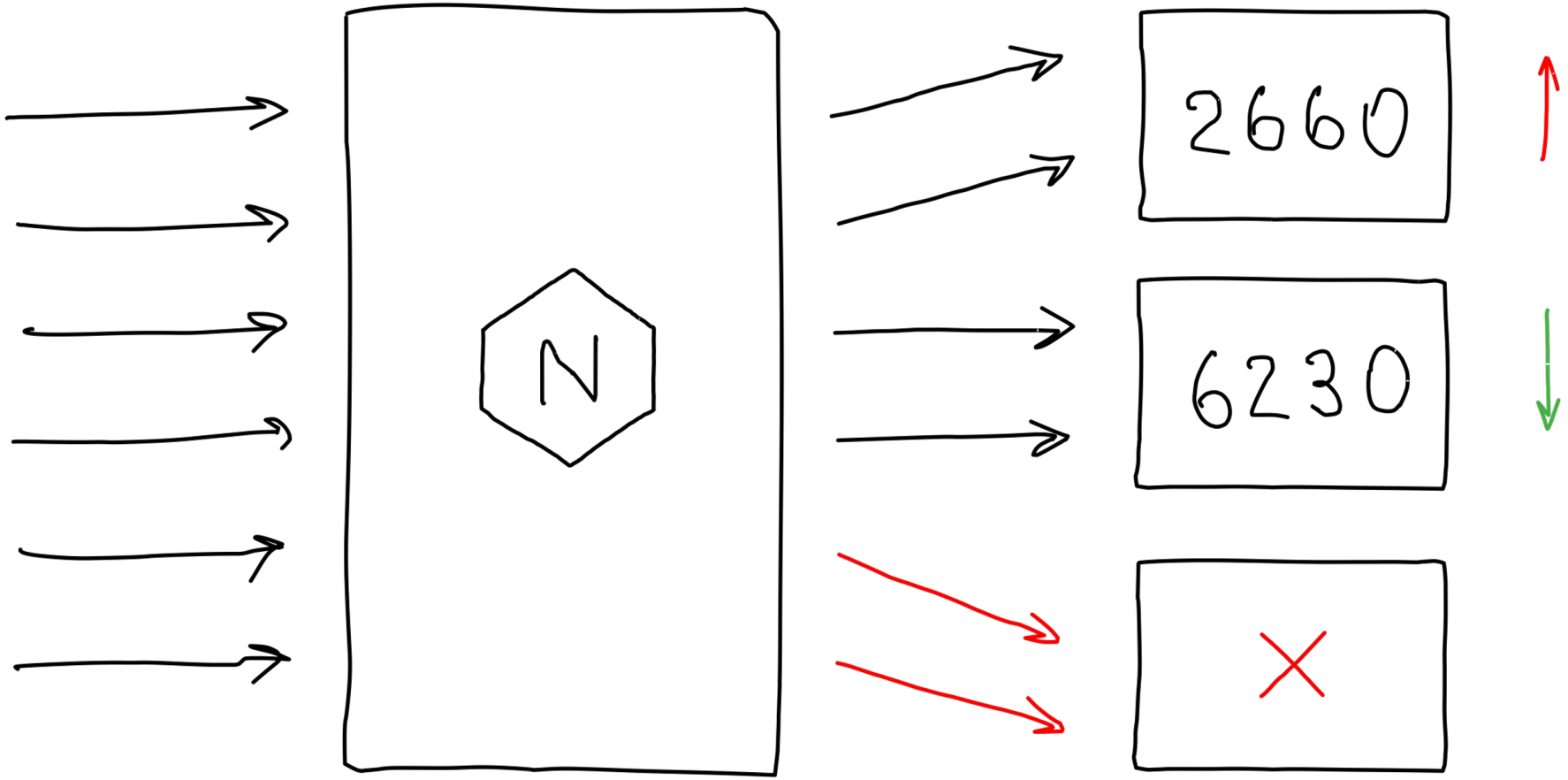
What are queues for?

- Task distribution

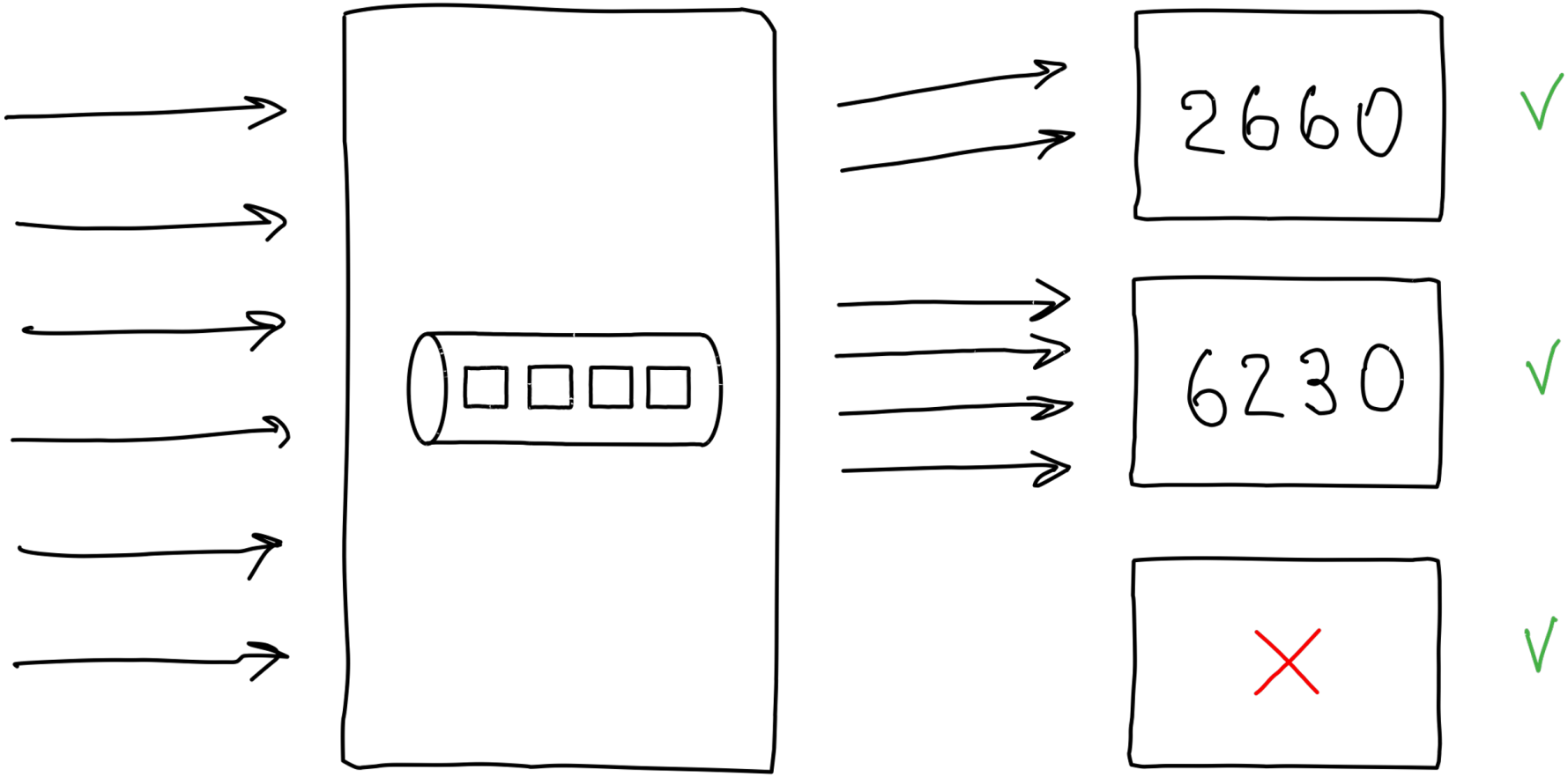
Task distribution



Task distribution



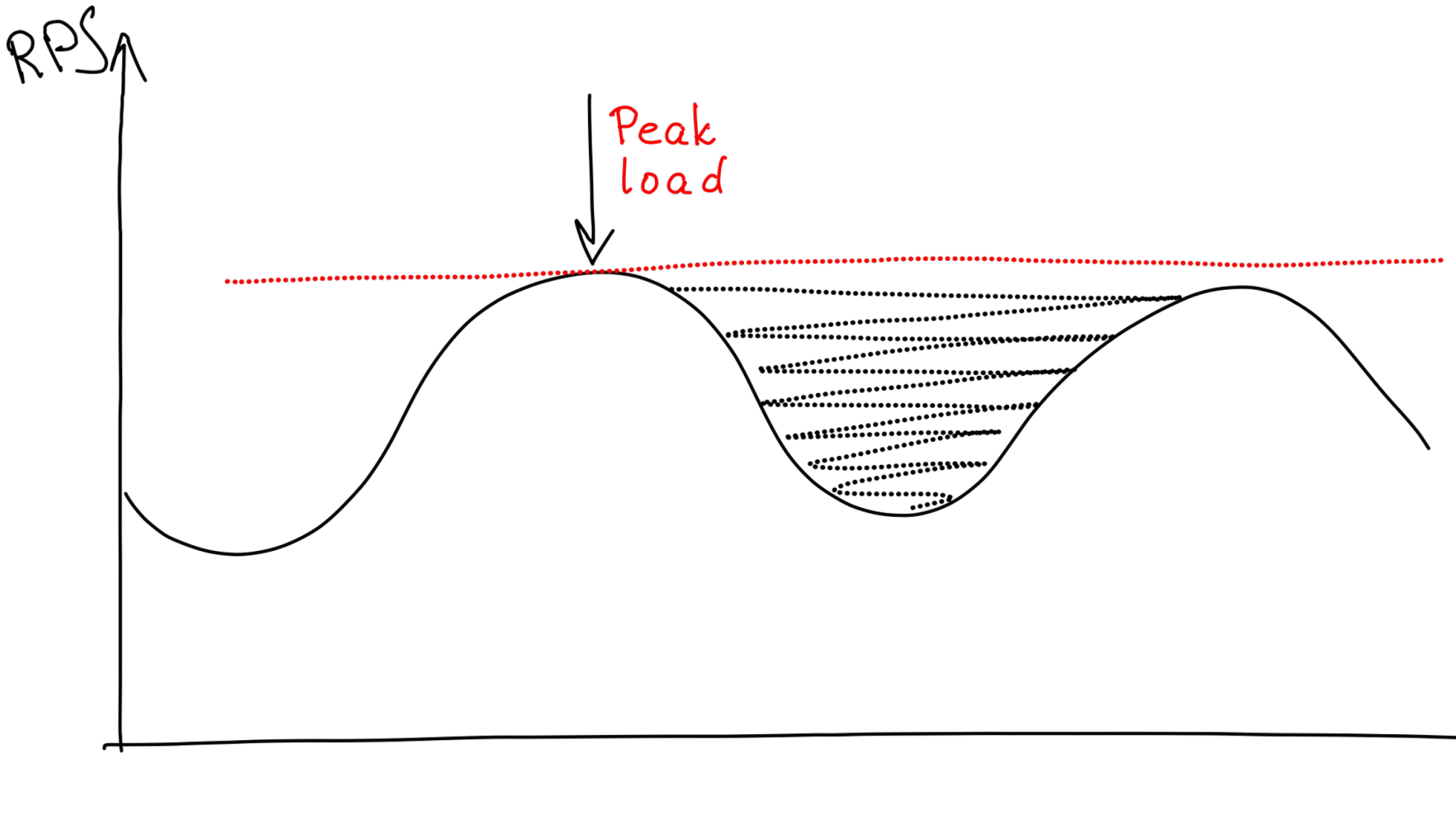
Task distribution



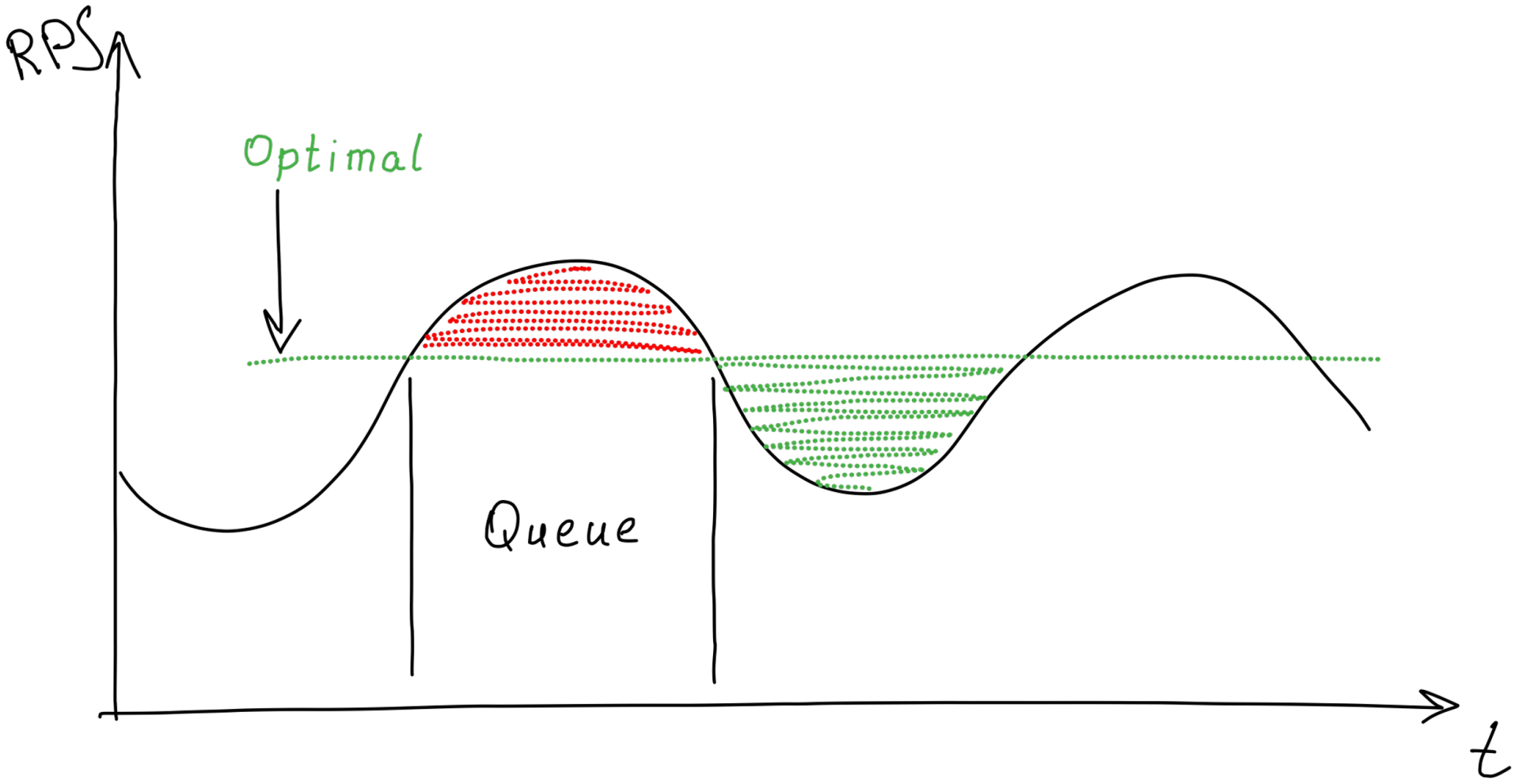
What are queues used for?

- Task distribution
- Execution scheduling

Execution scheduling



Execution scheduling



What are queues used for?

- Tasks distribution
- Scheduling of execution
- Fair resource allocation

What are queues used for?

- Tasks distribution
- Scheduling of execution
- Fair resource allocation

14:40—15:30, Today, 03 Hall

Fair threaded task scheduler verified in TLA+

Vladislav Shpilevoy
(Ubisoft)

What are queues used for?

- Tasks distribution
- Scheduling of execution
- Fair resource allocation
- Messages replication

What are queues used for?

- Tasks distribution
- Scheduling of execution
- Fair resource allocation
- Messages replication
- Fault tolerance, durability, delivery guarantee

What are queues used for?

- Tasks distribution
- Scheduling of execution
- Fair resource allocation
- Messages replication
- Fault tolerance, durability, delivery guarantee
- Microservices communication

What are queues used for?

- Tasks distribution
- Scheduling of execution
- Fair resource allocation
- Messages replication
- Fault tolerance, durability, delivery guarantee
- Microservices communication
- Event-driven architecture (Event Sourcing)
- Streaming architecture

Where are queues used?

- Hardware

IRQ
NCQ
Hardware Buffers

Where queues are used?

- Hardware
- OS Kernel

epoll / kqueue
networking
signal handling

Where queues are used?

- Hardware
- OS Kernel
- Applications

Cross thread
IPC

Where queues are used?

- Hardware
- OS Kernel
- Applications
- Network interactions

Where queues are used?

- Hardware
- OS Kernel
- Applications
- Network interactions
- Distributed systems

Where queues are used?

- Hardware
- OS Kernel
- Applications
- Network interactions
- Distributed systems
- Connection of businesses

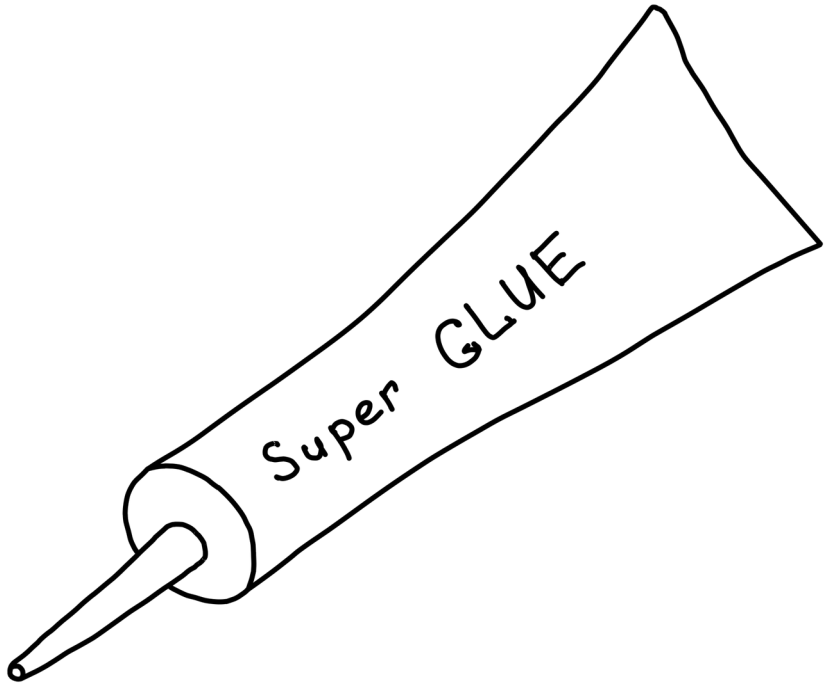
Where queues are used?

- Hardware
- OS Kernel
- Applications
- Network interactions
- Distributed systems
- Connection of businesses

- In fact — everywhere

Where queues are used?

- Hardware
- OS Kernel
- Applications
- Network interactions
- Distributed systems
- Connection of businesses



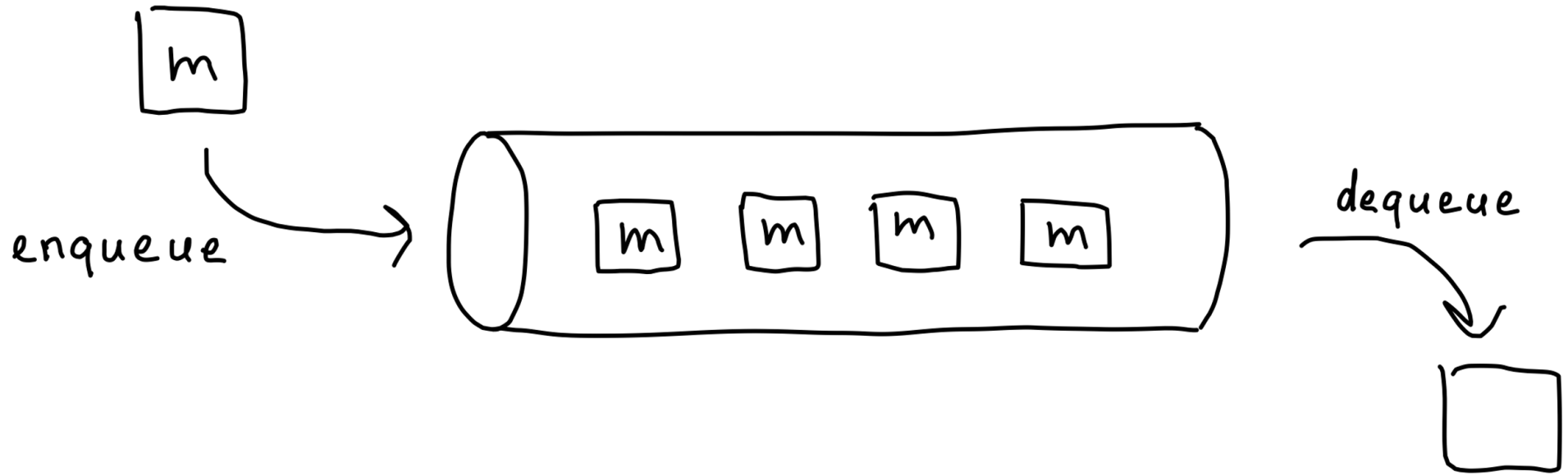
- In fact — everywhere

Queues are the glue

What is a queue?

- A media for communication using messages

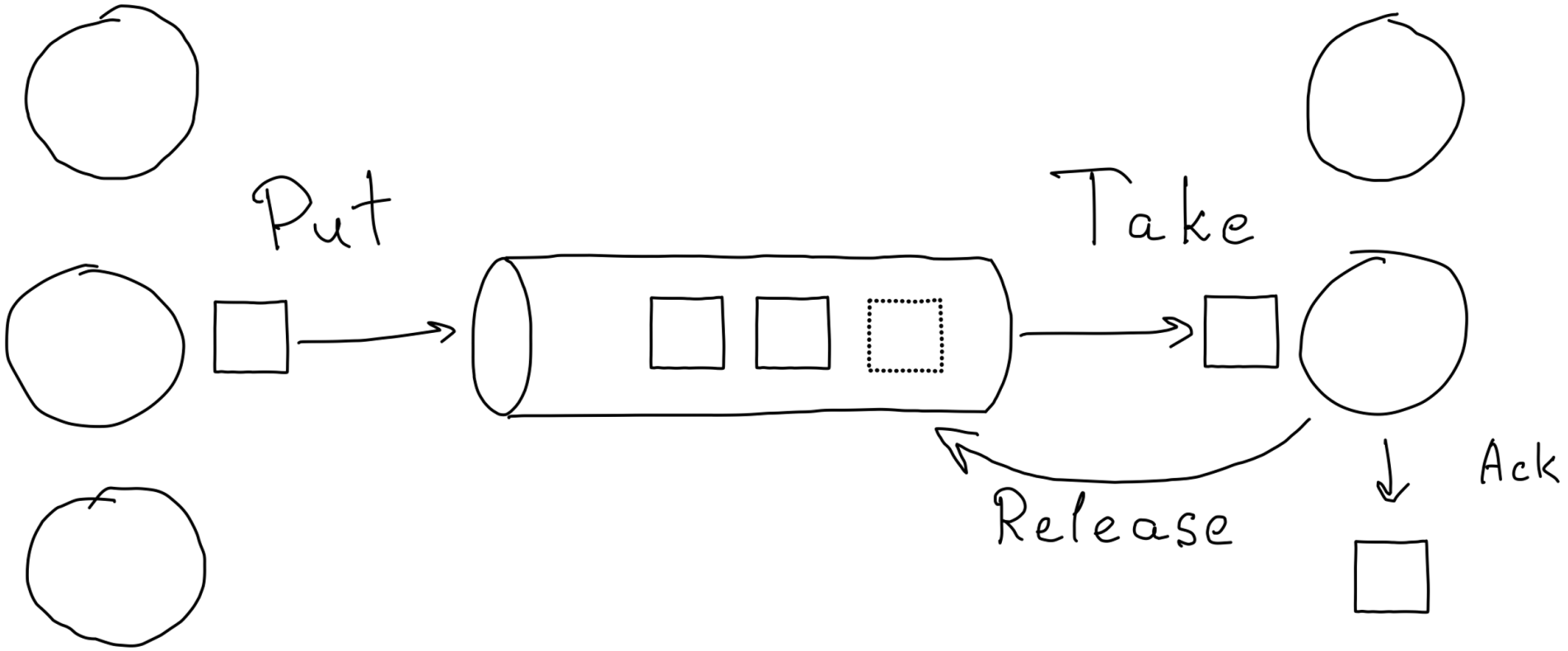
What is a queue?



What is a queue?

- A media for communication using messages
- One-to-one approach (Put/Take): 1 → 1

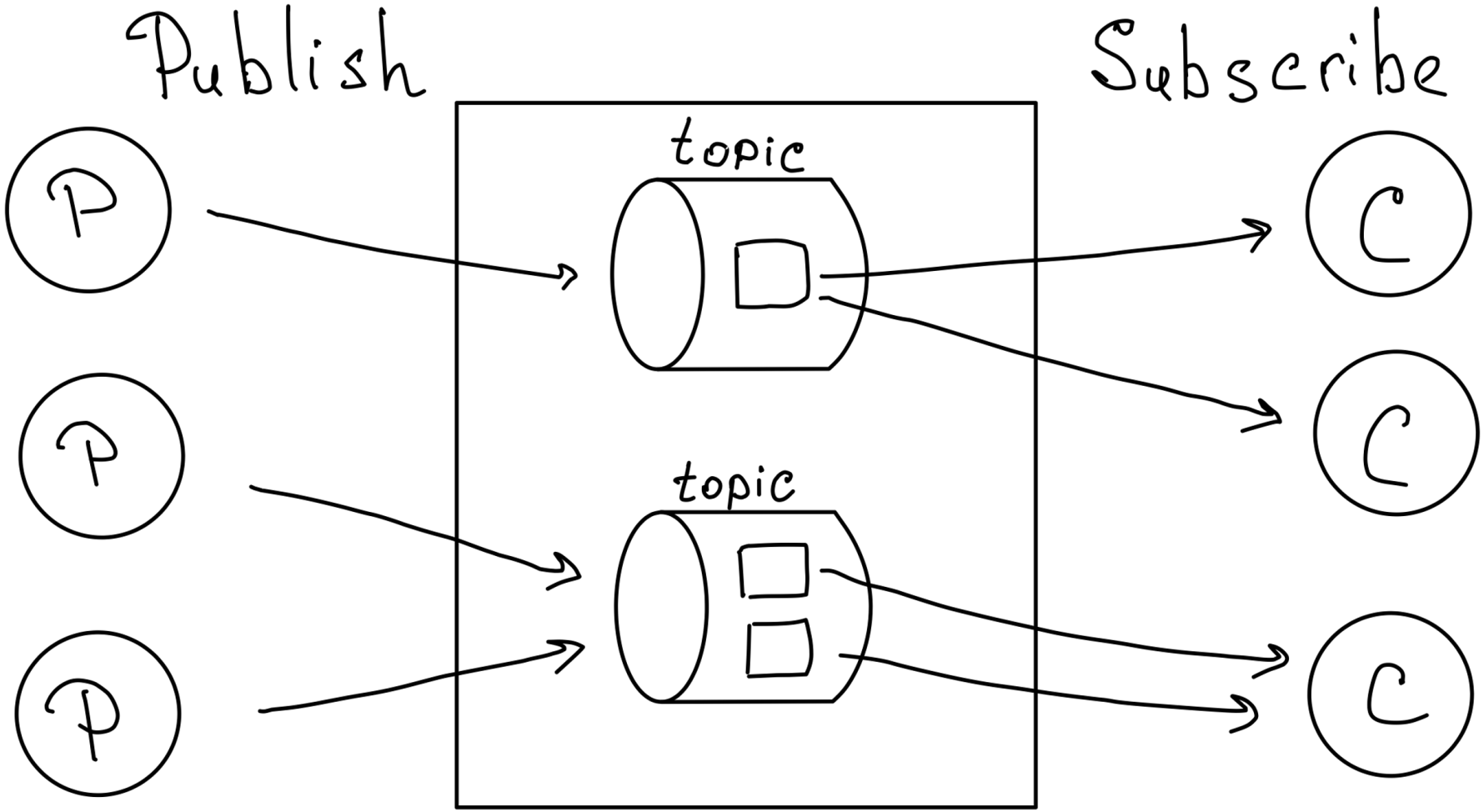
Put/Take



What is a queue?

- A media for communication using messages
- One-to-one approach (Put/Take): 1 → 1
- One-to-many approach (Publish/Subscribe): 1 → *

Pub/Sub



What is a queue?

- A media for communication using messages
- One-to-one approach (Put/Take): 1 → 1
- One-to-many approach (Publish/Subscribe): 1 → *
- Synchronous flow (Request/Response): 1 ⇆ 1

What is a queue?

- A media for communication using messages
- One-to-one approach (Put/Take): 1 → 1
- One-to-many approach (Publish/Subscribe): 1 → *
- Synchronous flow (Request/Response): 1 ⇆ 1
- Protocols: AMQP, MQTT, STOMP, NATS, ZeroMQ, ...

What are the options?

- Cloud solutions
  - Amazon SQS, EventBridge
  - Google Cloud Tasks
  - CloudAMQP
  - …

What are the options?

- Cloud solutions
  - Amazon SQS, EventBridge, Google Cloud Tasks, CloudAMQP, ...
- Message brokers
  - RabbitMQ
  - Apache Kafka
  - ActiveMQ
  - Tarantool Queue
  - NATS
  - NSQ

What are the options?

- Cloud solutions
  - Amazon SQS, EventBridge, Google Cloud Tasks, CloudAMQP, ...
- Message brokers
  - RabbitMQ, Apache Kafka, Tarantool Queue, NATS, NSQ, ...
- Database-backed queues
  - PgQueue
  - Tarantool
  - Redis

What are the options?

- Database-backed queues
  - PgQueue
  - Tarantool
  - Redis

14:40—15:30, Tomorrow, 01 Hall
Transactional queues in PostgreSQL

Igor Loban
(Toloka.ai)

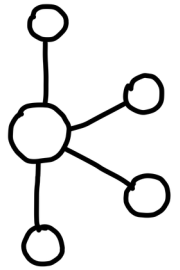
What are the options?

- Cloud solutions
  - Amazon SQS, EventBridge, Google Cloud Tasks, CloudAMQP, ...
- Message brokers
  - **RabbitMQ**, Apache **Kafka**, Tarantool Queue, **NATS**, NSQ, ...
- Database-backed queues
  - PgQueue, Tarantool, Redis, ...
- “Sockets on steroids”
  - NATS, ZeroMQ

Who are the players?

- Apache Kafka
  - Distributed message log for streaming
- RabbitMQ
  - Traditional broker with AMQP protocol
- Managed Cloud Queue (SQS/EventBridge/MQ/...)
  - The solution to be used in clouds
- NATS
  - Microservices communication bridge
- Tarantool
  - The platform for custom built solutions

Apache Kafka



- Replicated & sharded message log
- Strict delivery order (FIFO)
- A limited number of consumers
- Stream replay
- Apache ecosystem integration
- Mainly used for
  - Data analysis. Logs, metrics, audit
  - High performance stream data processing
  - Consistent data replication

Apache Kafka

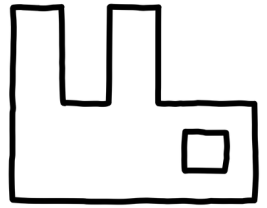
11:10—12:00, Tomorrow, 01 Hall
Kafka architecture: perormance

Alexander Sibiryakov (Zyte)

15:50—16:40, Tomorrow, 02 Hall
Kafka for Golang delevelopers: tips and tricks

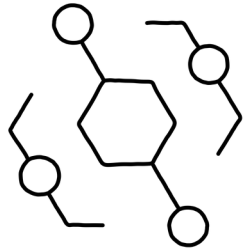
Denis Filippov (Coins.ph)

RabbitMQ



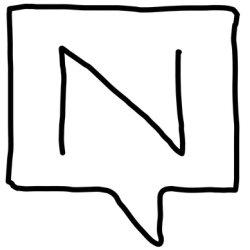
- Protocols: AMQP, MQTT, STOMP
- Prioritized, delayed and background tasks
- Unlimited number of consumers
- Flexible storage: memory, disk, replication, quorum
- Easy to learn. Hard to master
- Mainly used for
  - Microservices communication
  - As a message bus
  - As a traditional pub/sub broker

Managed Cloud Queue



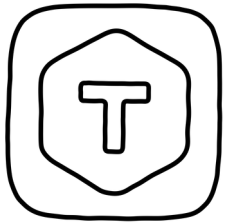
- Reliable and scalable queue
- Stateless protocol, simple interaction
- Standard API
- Pay-as-you-go: low consumption — low expenses
- Mainly used for
  - Communication between cloud components
  - Serverless architecture

NATS Messaging



- Fast non-persistent message communication
- High performance and scalability
- Multi-paradigm: pub/sub, put/take, req/res
- Expandable with JetStream
  - Streaming support
  - Durable storage, RAFT cluster
- Mainly used as
  - Messaging media for distributed systems

Tarantool



- High-performance broker (Tarantool Queue)
- Integration with streaming systems
- The platform for building custom queues
- Arbitrary logic and event processing
- Transaction support within exchange
- Mainly used for
  - High performance and high throughput message delivery
  - Building complex event processing with custom logic

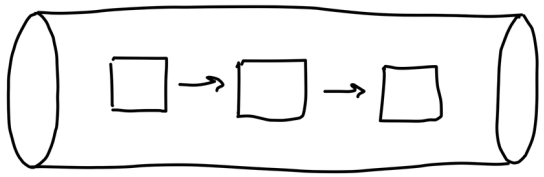
Possible problems

Or what is worth to know?

- Algorithms of queues
- Network problems
- “Exactly once”
- Network (again) and disk problems
- Outages & failures

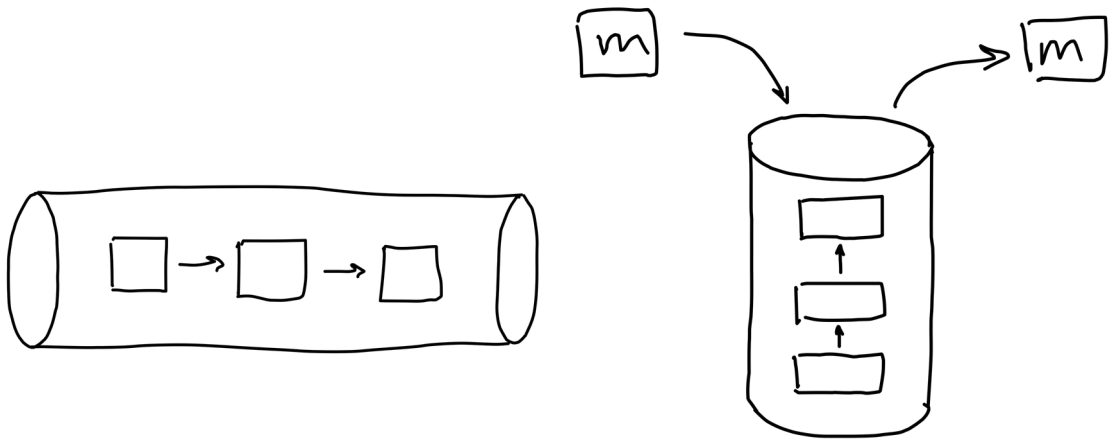
Algorithms

- FIFO, LIFO, Best Effort, QoS



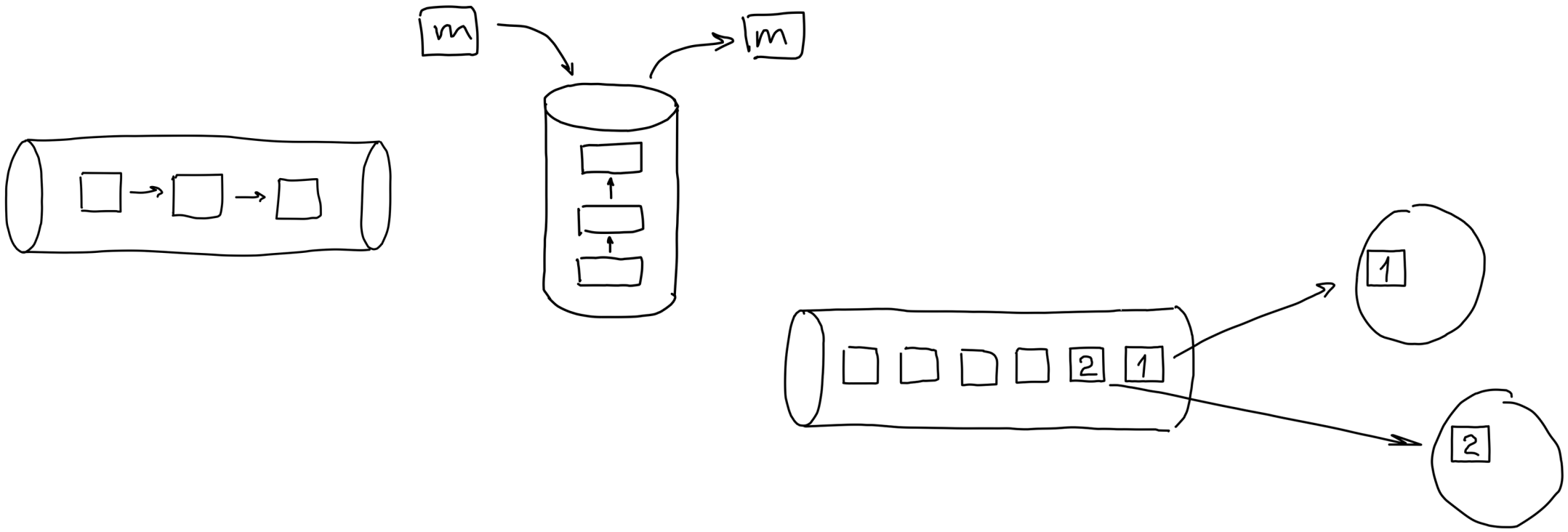
Algorithms

- FIFO, LIFO, Best Effort, QoS



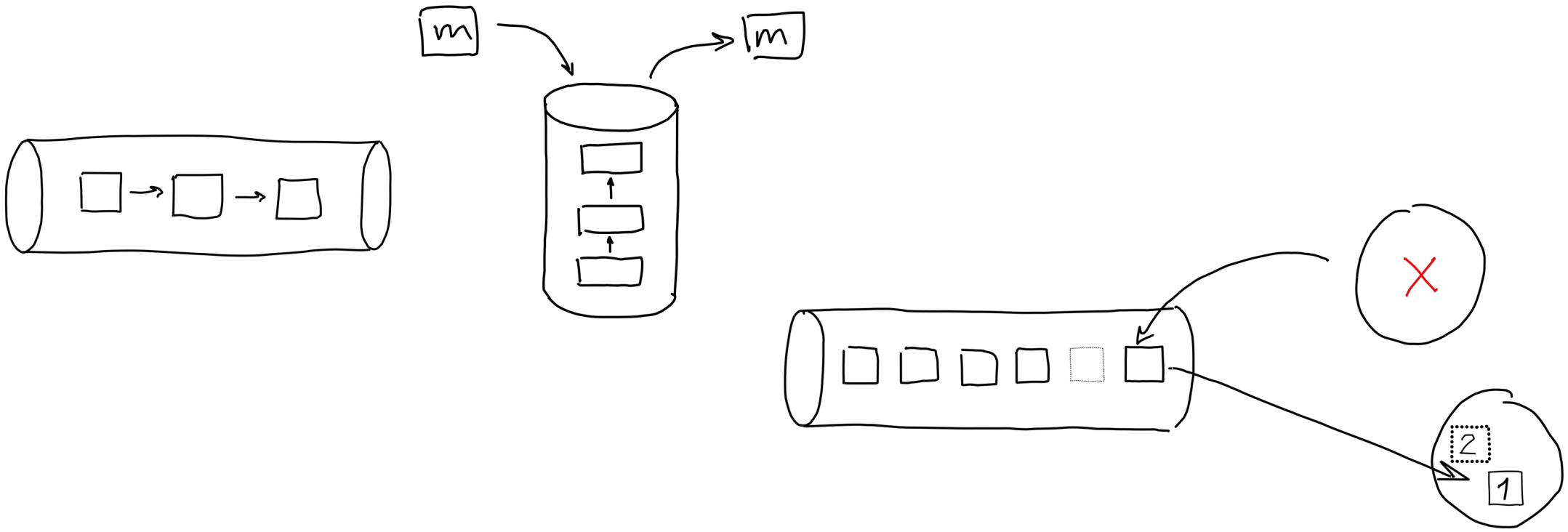
Algorithms

- FIFO, LIFO, Best Effort, QoS



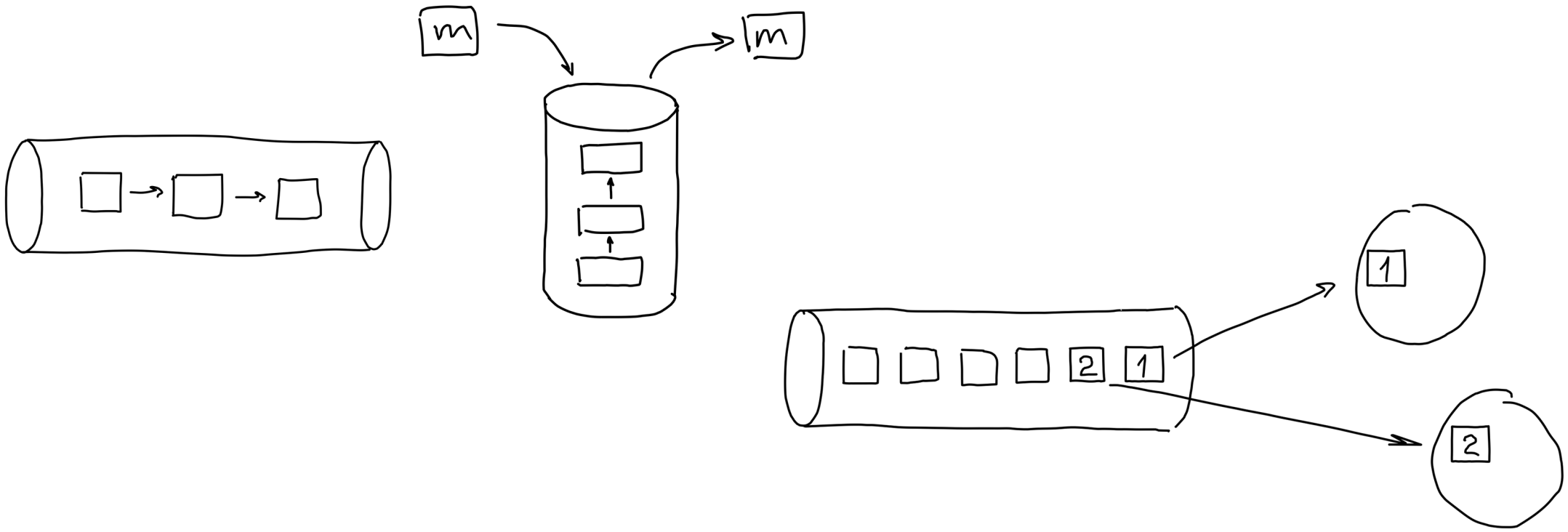
Algorithms

- FIFO, LIFO, Best Effort, QoS



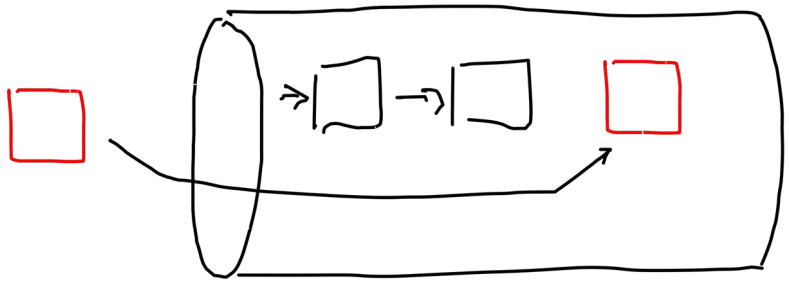
Algorithms

- FIFO, LIFO, Best Effort, QoS



Algorithms

- FIFO, LIFO, Best Effort, QoS
- Message prioritization



Algorithms

- FIFO, LIFO, Best Effort, QoS
- Message prioritization
- Nested queues, hierarchy

Algorithms

- FIFO, LIFO, Best Effort, QoS
- Message prioritization
- Nested queues, hierarchy
- Retry, delay, retry with delay

Algorithms

- FIFO, LIFO, Best Effort, QoS
- Message prioritization
- Nested queues, hierarchy
- Retry, delay, retry with delay
- Dead letter queue (and reordering)

Algorithms

- FIFO, LIFO, Best Effort, QoS
- Message prioritization
- Nested queues, hierarchy
- Retry, delay, retry with delay
- Dead letter queue (and reordering)
- Task dependency

Algorithms

- FIFO, LIFO, Best Effort, QoS
- Message prioritization
- Nested queues, hierarchy
- Retry, delay, retry with delay
- Dead letter queue (and reordering)
- Task dependency
- TTL, TTR, Putback

Would you like yet more?

- Prioritization and Starvation

Would you like yet more?

- Prioritization and Starvation
- Throughput
- Performance

Would you like yet more?

- Prioritization and Starvation
- Throughput
- Performance
- Scalability

Would you like yet more?

- Prioritization and Starvation
- Throughput
- Performance
- Scalability
- Limits and Capacity

Would you like yet more?

- Prioritization and Starvation
- Throughput
- Performance
- Scalability
- Limits and Capacity
- Reliability

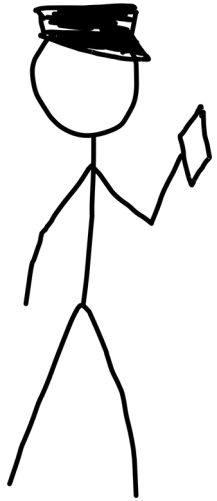
Would you like yet more?

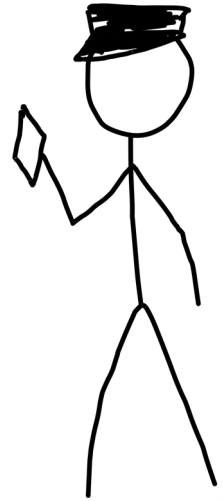
- Prioritization and Starvation
- Throughput
- Performance
- Scalability
- Limits and Capacity
- Reliability → Availability + Durability

What’s next? Network

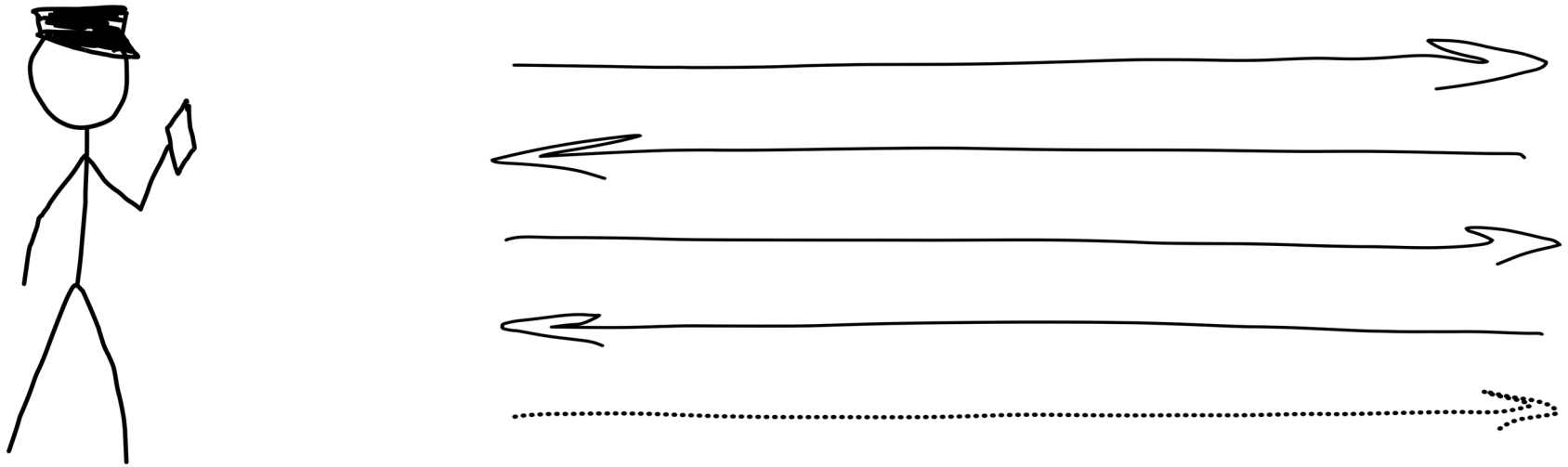
- Undefined behavior

Two Generals' Problem

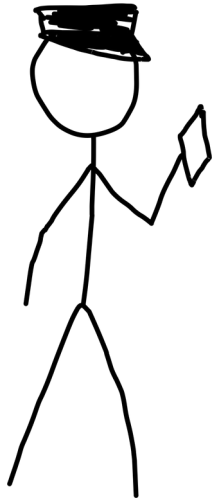




Two Generals' Problem

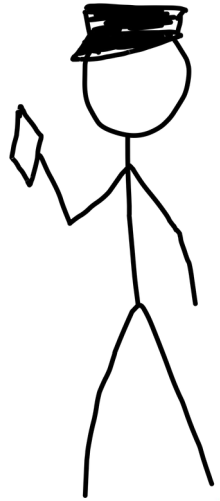


Two Generals' Problem

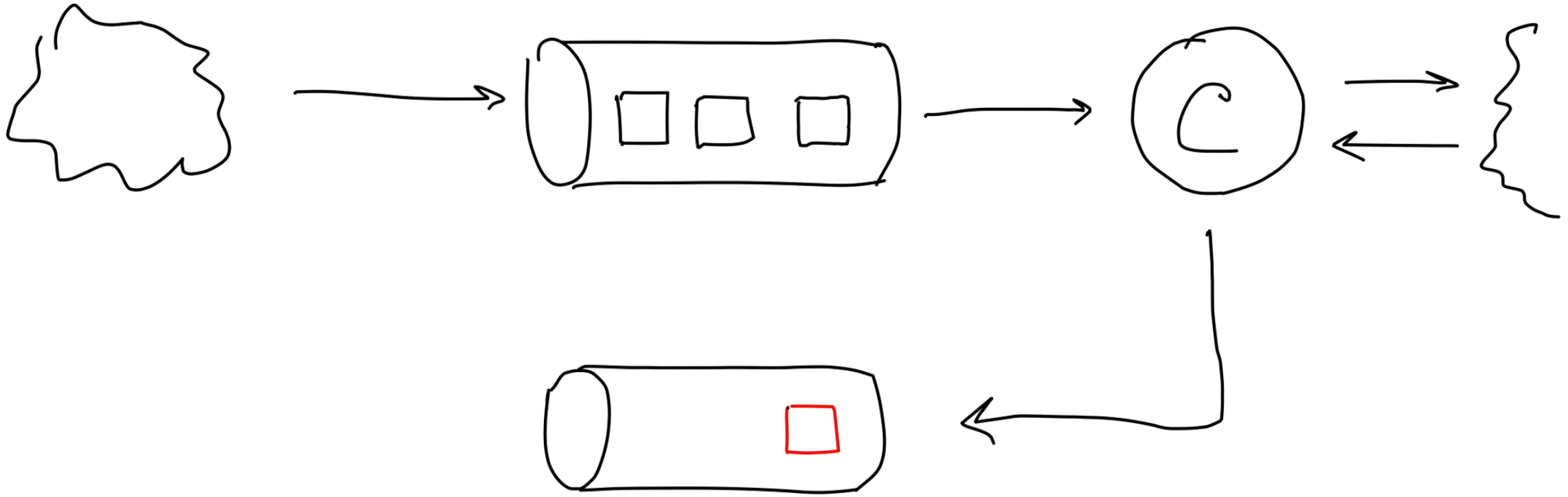


There are two complex problems in distributed systems:

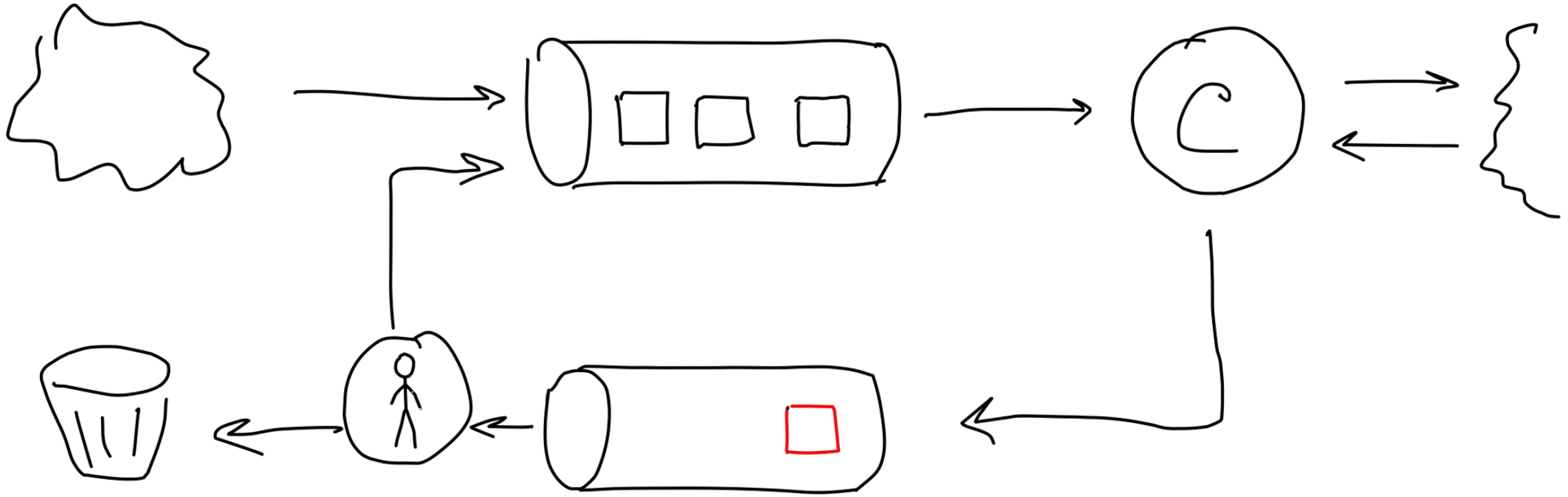
2. Exactly once delivery
    1. Strict message order
2. Exactly once delivery



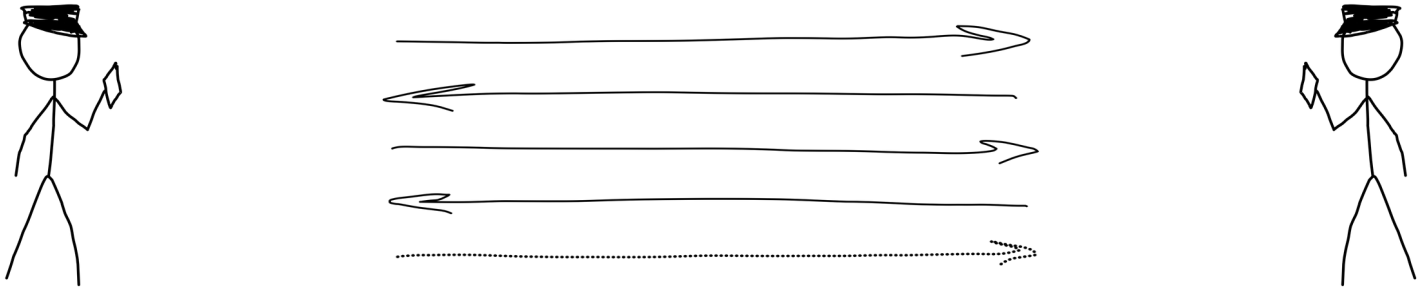
Solution for “exactly once”



Solution for “exactly once”



Solution for “exactly once”



Problems: network & disk

- Throughput
- Latency

Outages & failures

- Hardware failures
  - Disk
  - Host
  - Datacenter

Outages & failures

- Hardware failures
  - Disk
  - Host
  - Datacenter
- Temporary failures
  - Power
  - Network
  - Split brain

Outages & failures

- Hardware failures
  - Disk
  - Host
  - Datacenter
- Temporary failures
  - Power
  - Network
  - Split brain
- Permanent failure
  - Physical destruction

Outages & failures

- Hardware failures
  - Disk
  - Host
  - Datacenter
- Temporary failures
  - Power
  - Network
  - Split brain
- Permanent failure
  - Physical destruction

Reliability explained

- Availability

- Durability

Reliability explained

- Availability
  - Ability to receive messages
  - Ability to store messages
- Durability

Reliability explained

- Availability
  - Ability to receive messages
  - Ability to store messages
- Durability
  - Ability to not lose received messages
  - Delivery guarantee

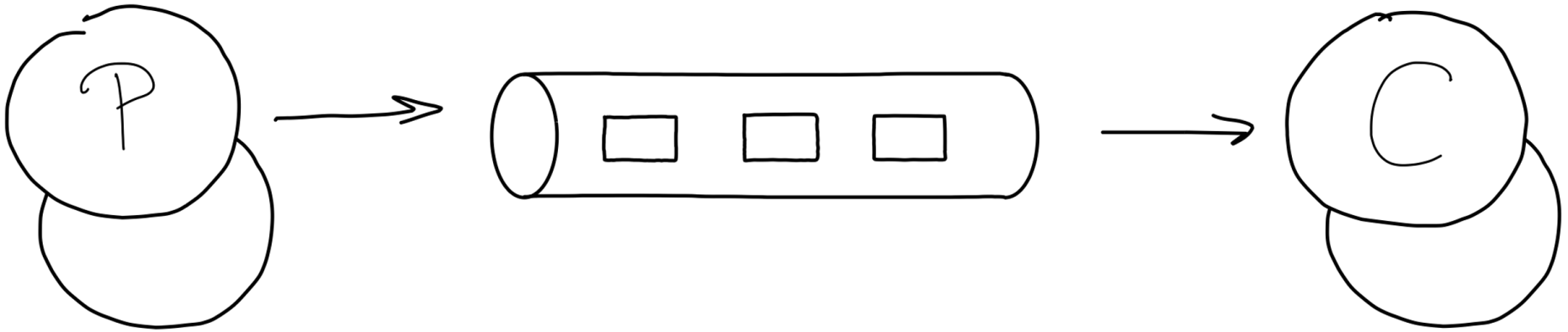
Main queue properties

- Availability
  - Ability to receive & store messages
- Durability
  - Ability to not lose received messages
- Delivery guarantee
  - A promise to deliver
  - “At least once” ($X \geq 1$)
  - “At most once” ($X \leq 1$)
  - “Exactly once” is impossible, while stated by many*

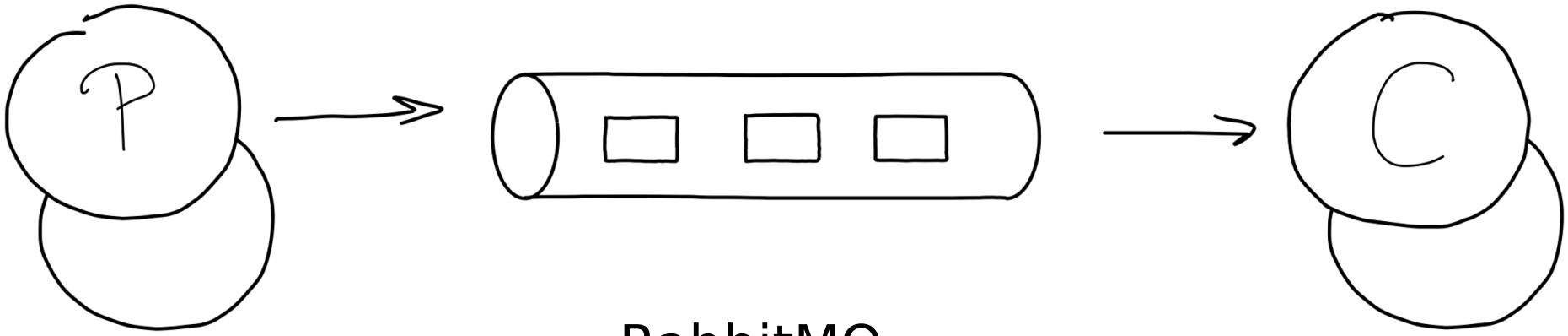
Main queue properties

- Availability
  - Ability to receive & store messages
- Durability
  - Ability to not lose received messages
- Delivery guarantee
  - A promise to deliver “At least once” or “At most once”
- Scalability
  - Ability to increase throughput by adding nodes

Topology: single instance

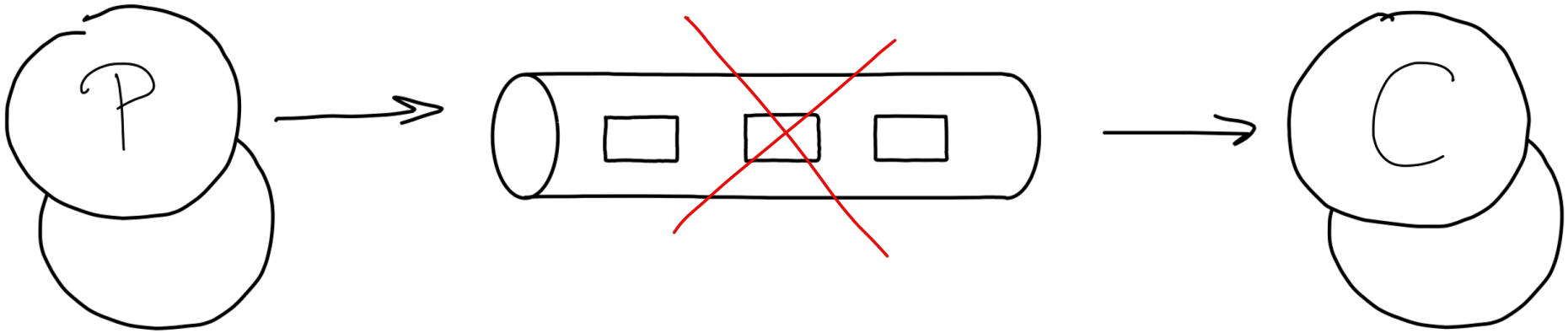


Topology: single instance



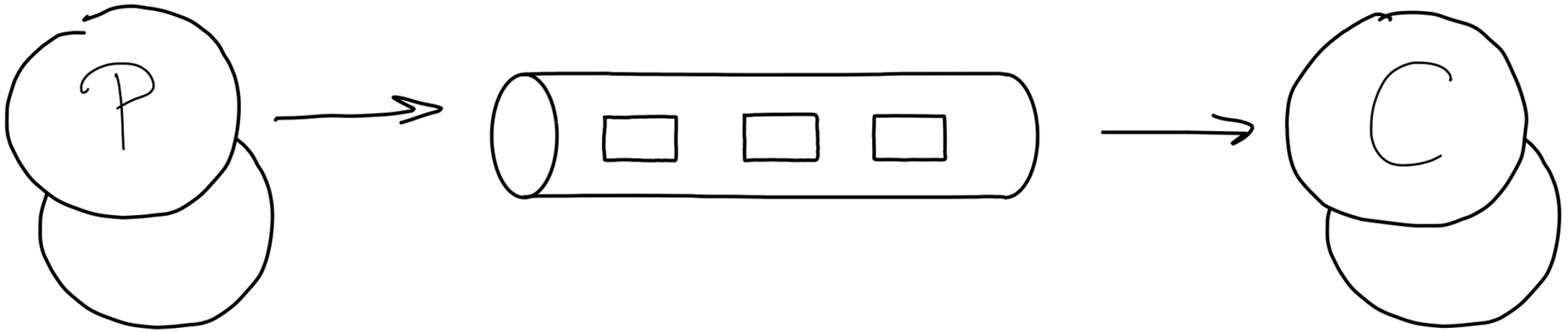
RabbitMQ
Redis
ActiveMQ
Beanstakld
...

Topology: single instance



Availability: low
Durability: low

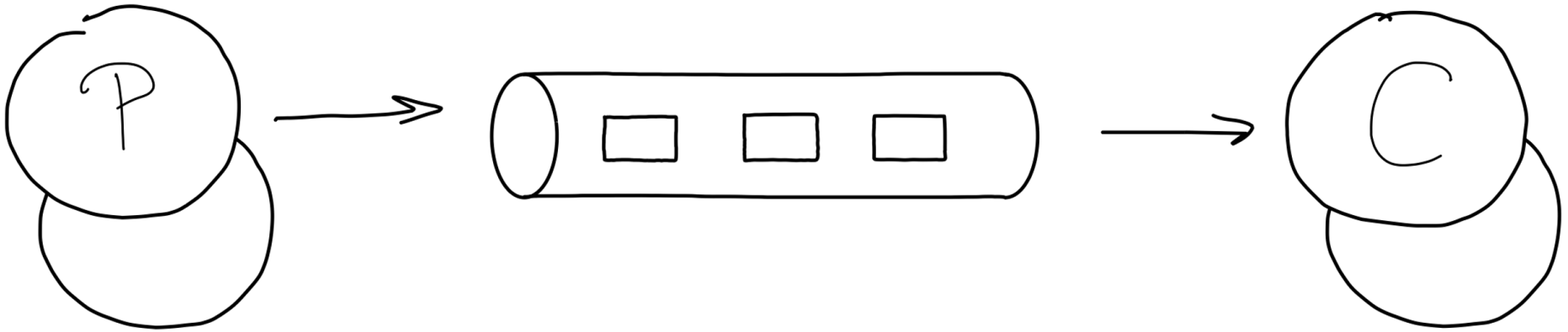
Topology: single instance



Scalability: no

Availability: low
Durability: low

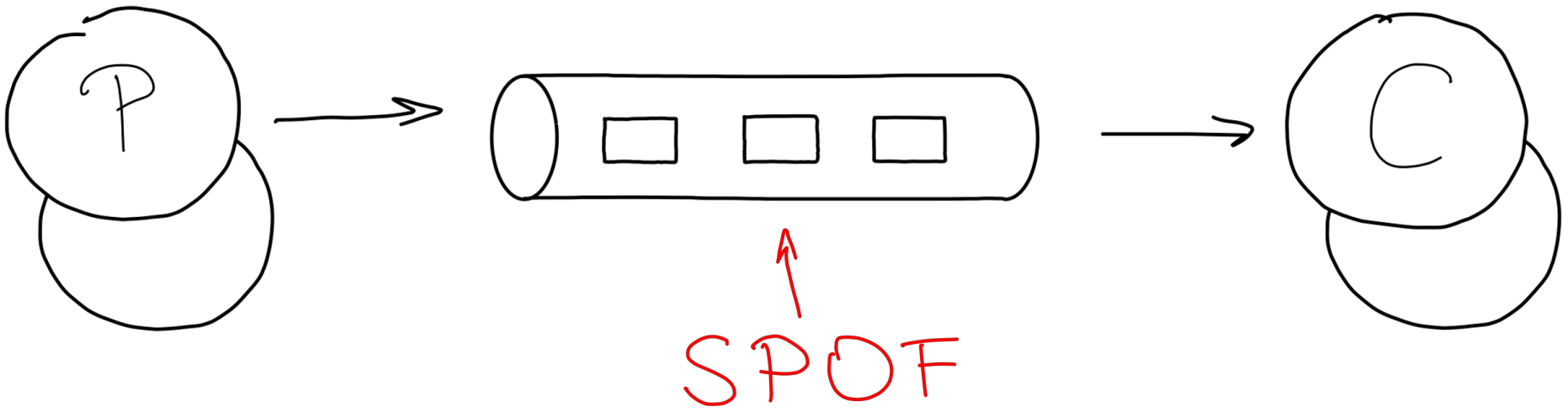
Topology: single instance



Scalability: no
Guarantee: X ≤ 1, X ≥ 1

Availability: low
Durability: low

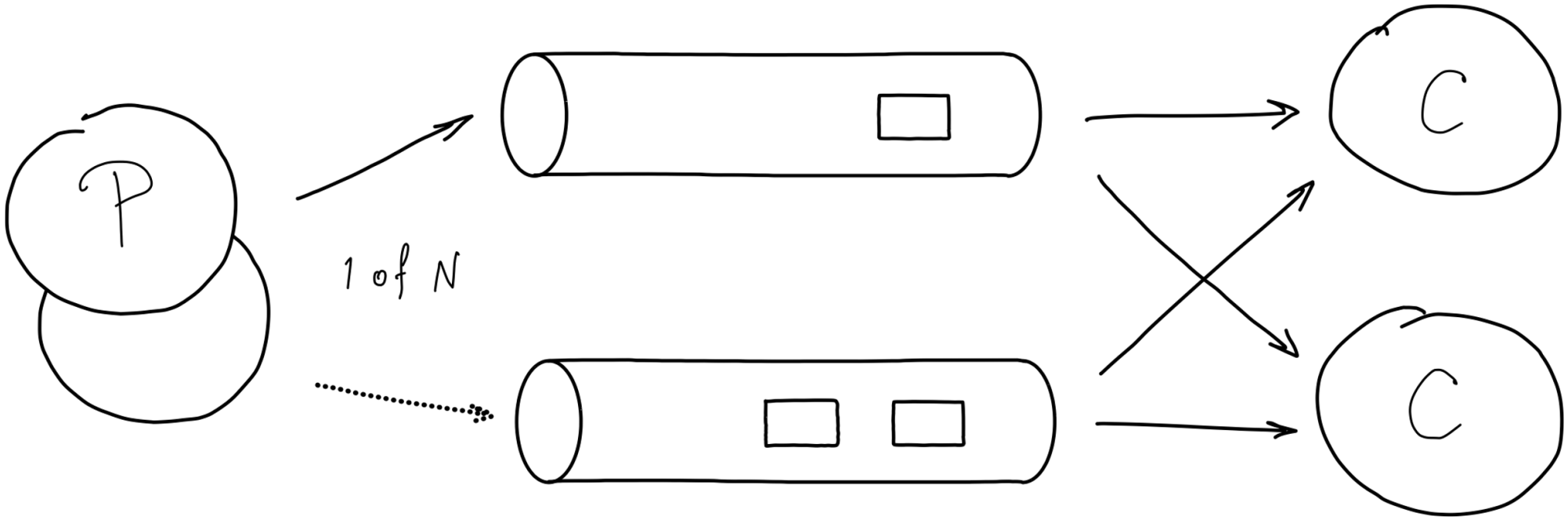
Topology: single instance



Scalability: no
Guarantee: X ≤ 1, X ≥ 1

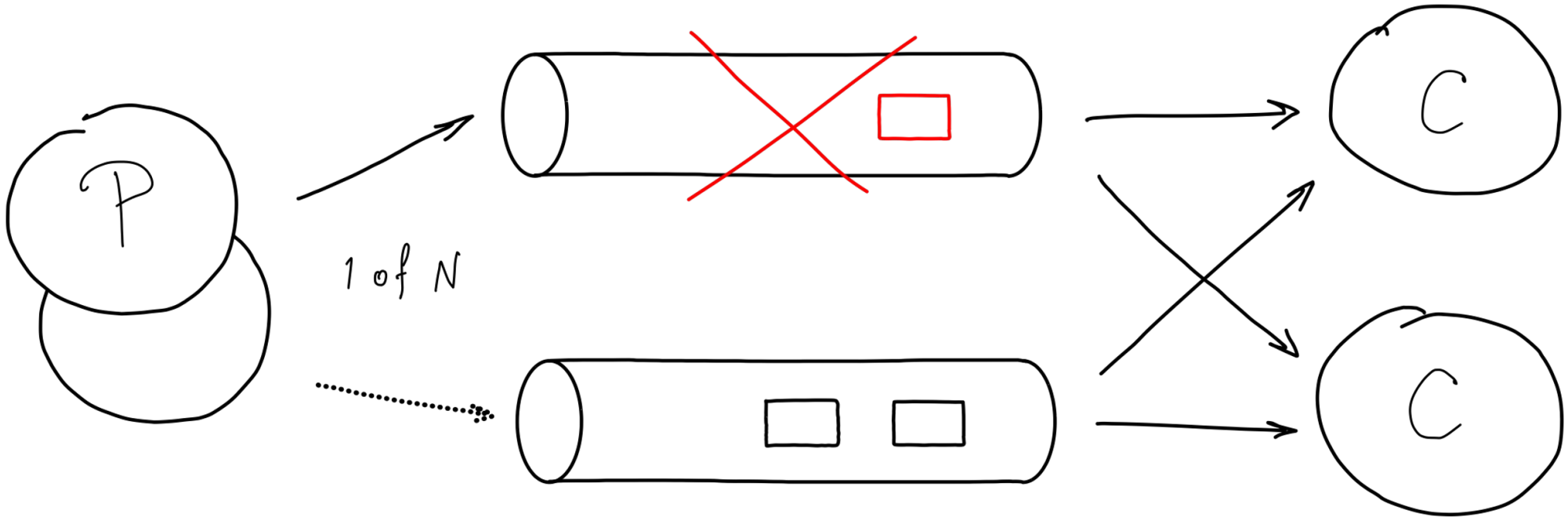
Availability: low
Durability: low

Topology: multi-instance



Scalability: yes

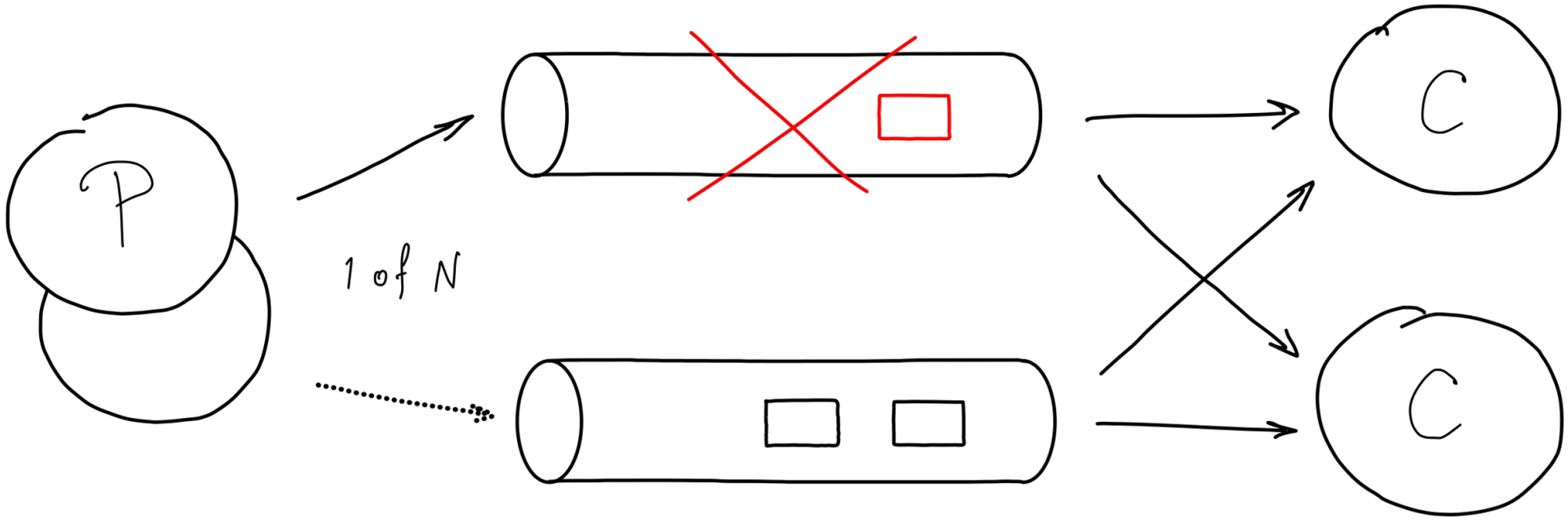
Multiple queues, put to 1



Scalability: yes

Availability: high

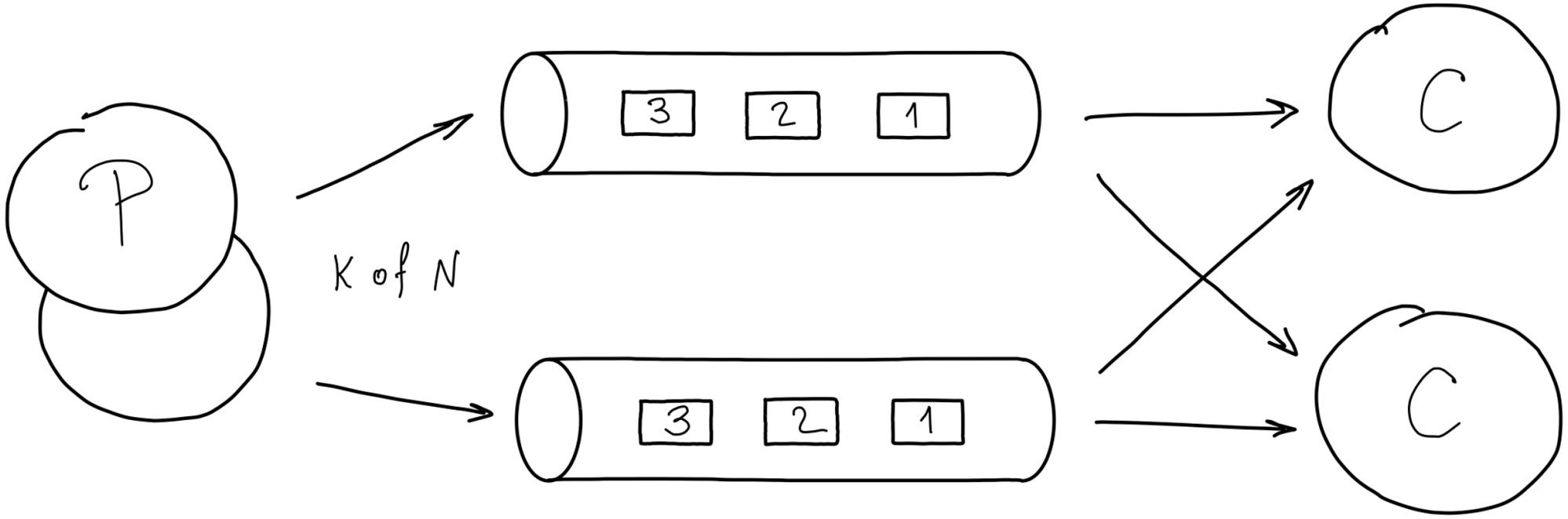
Multiple queues, put to 1



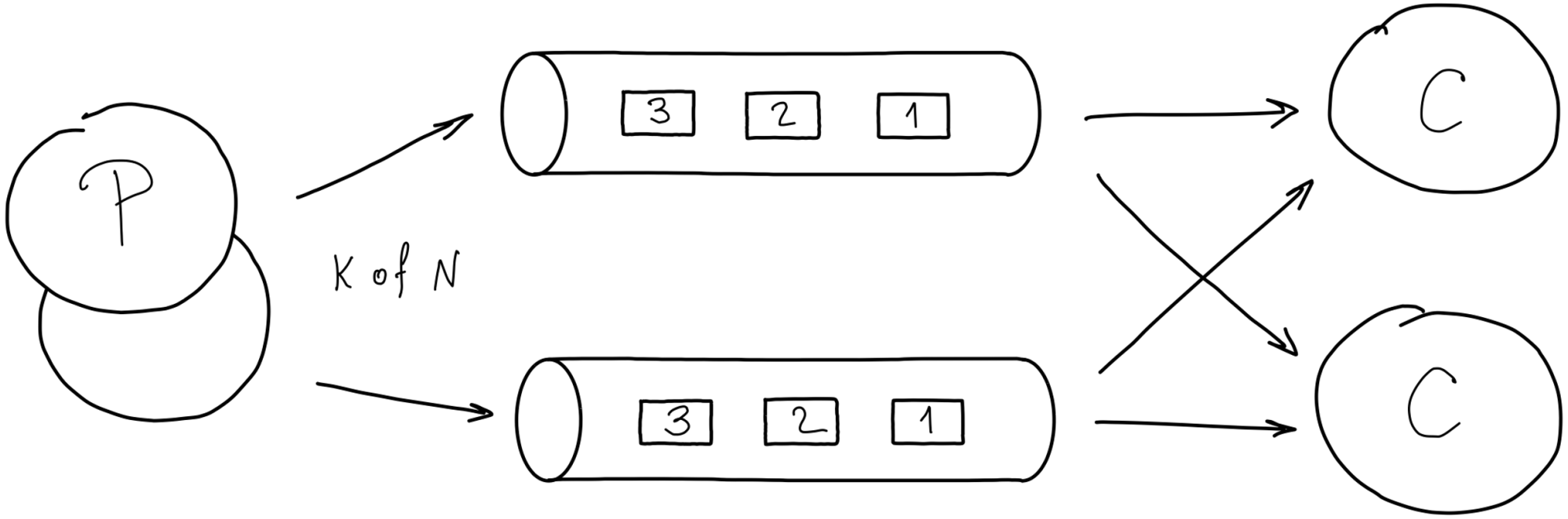
Scalability: yes
Guarantee: X ≤ 1, X ≥ 1

Availability: high
Durability: medium

Multiple queues, put to K/N

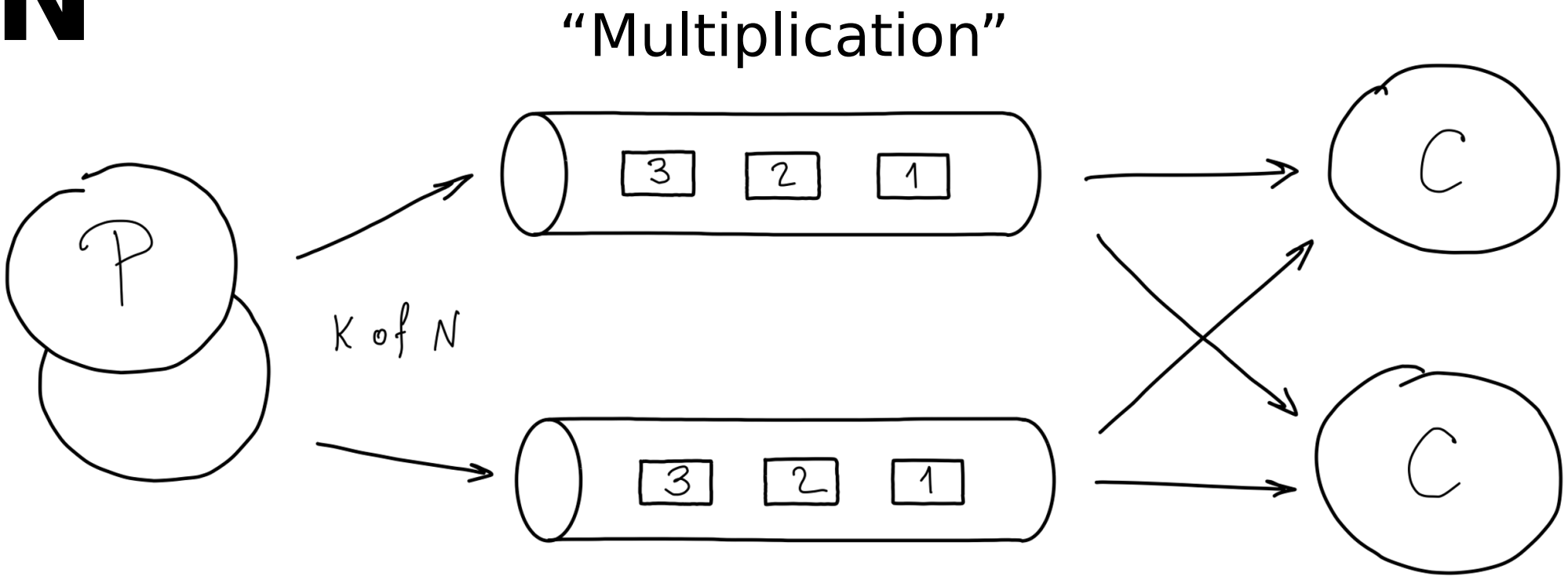


Multiple queues, put to K/N



Availability: high
Durability: high

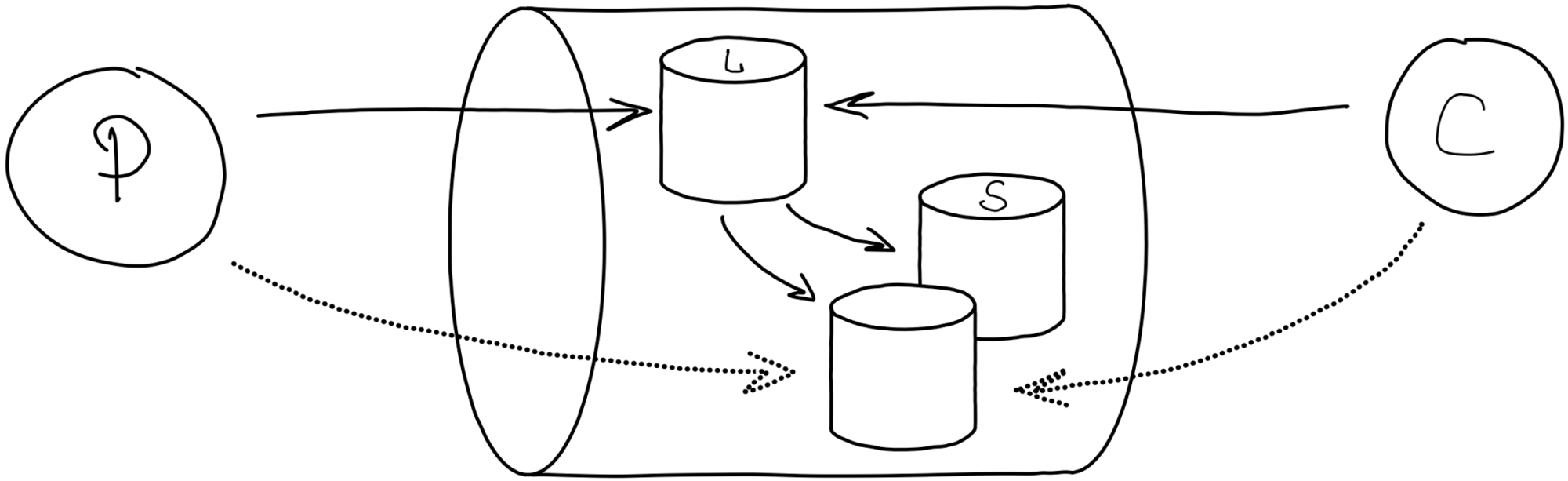
Multiple queues, put to K/N



Scalability: yes
Guarantee: X ≥ K

Availability: high
Durability: high

Replication

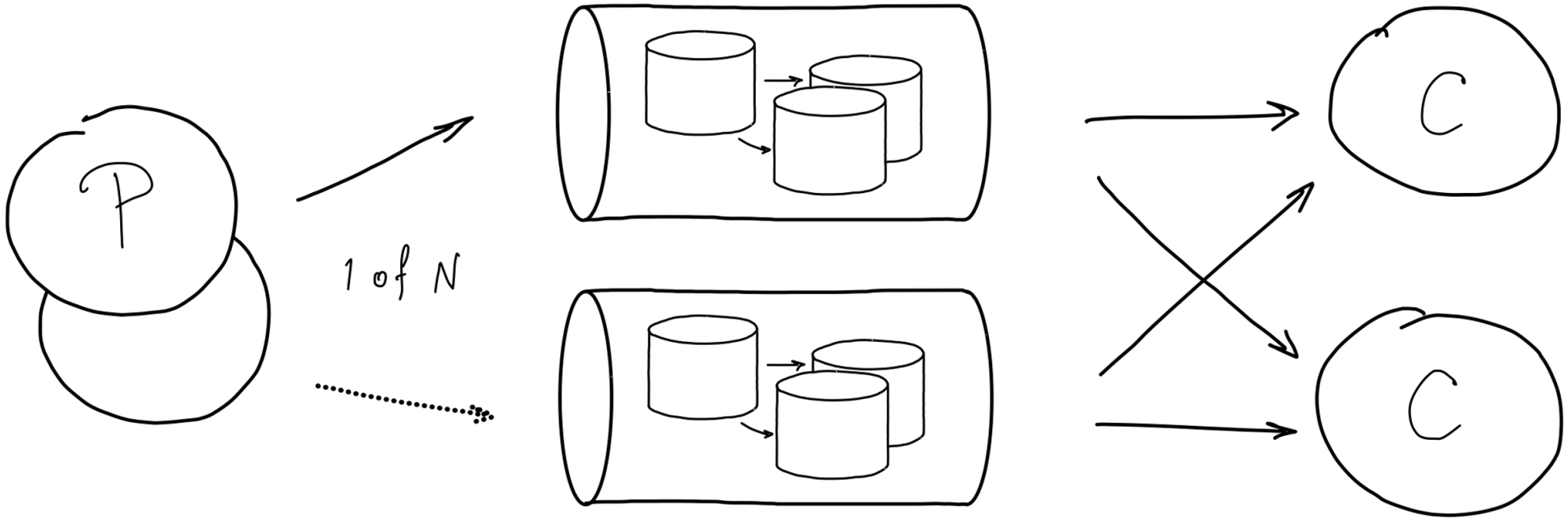


Interaction
with a leader

Replicas
are on standby

Replicated queues, 1/N

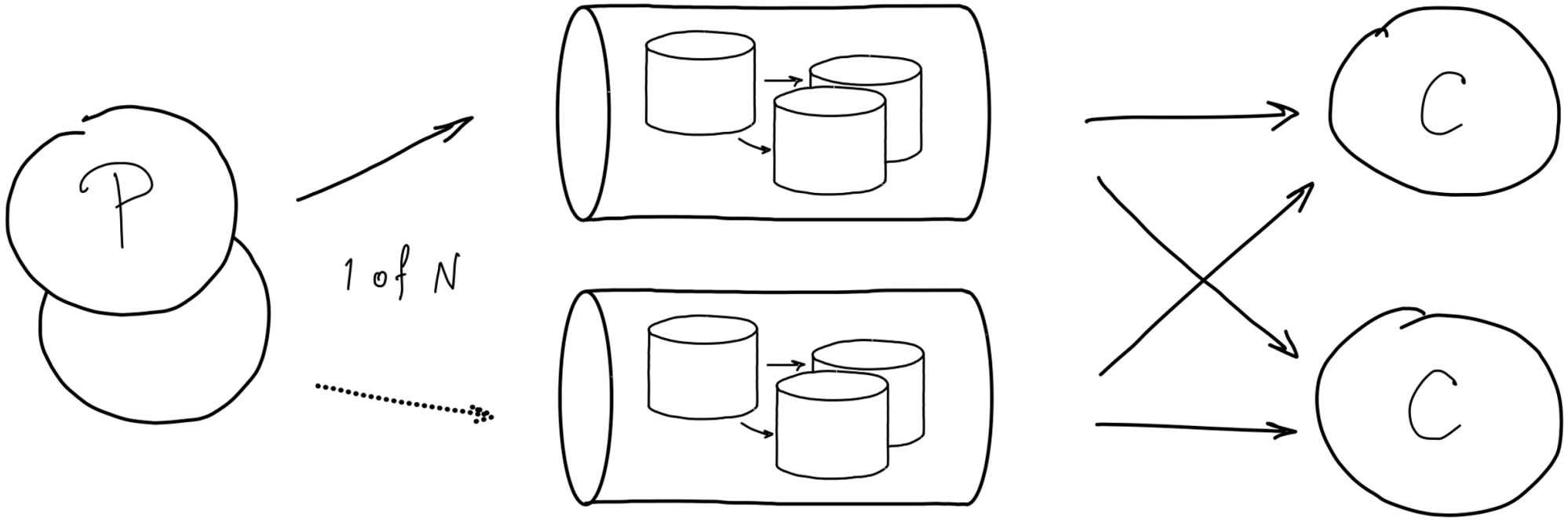
“like a database”



Availability: high
Durability: high

Replicated queues, 1/N

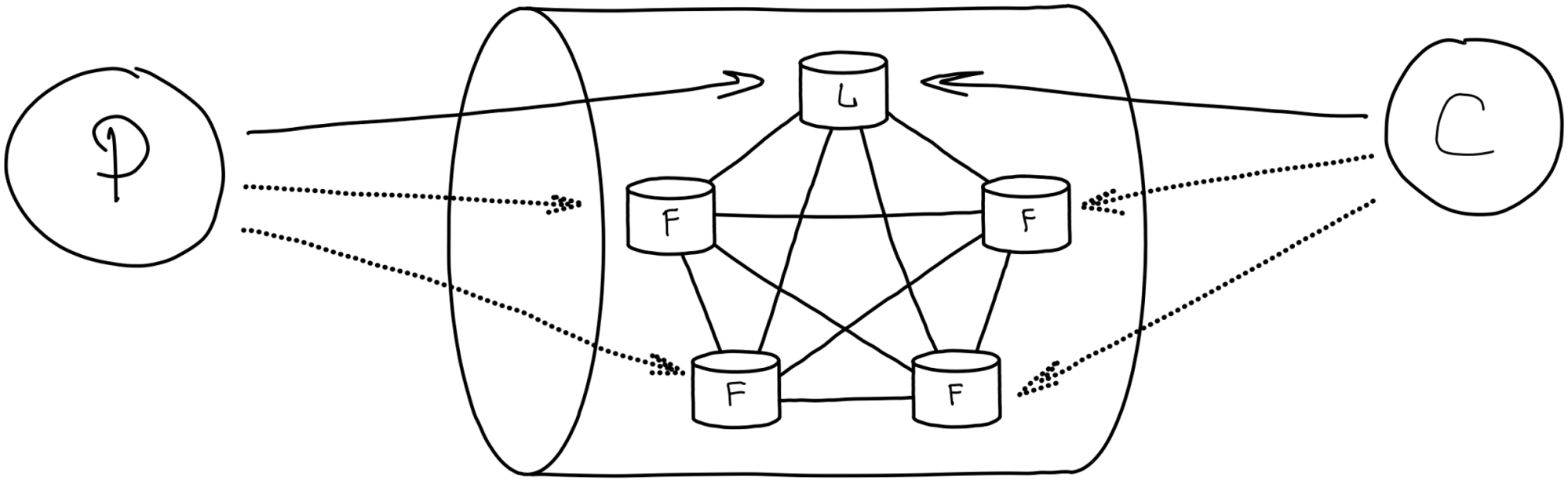
“like a database”



Scalability: yes
Guarantee: X ≈ 1 (X ≥ 1)

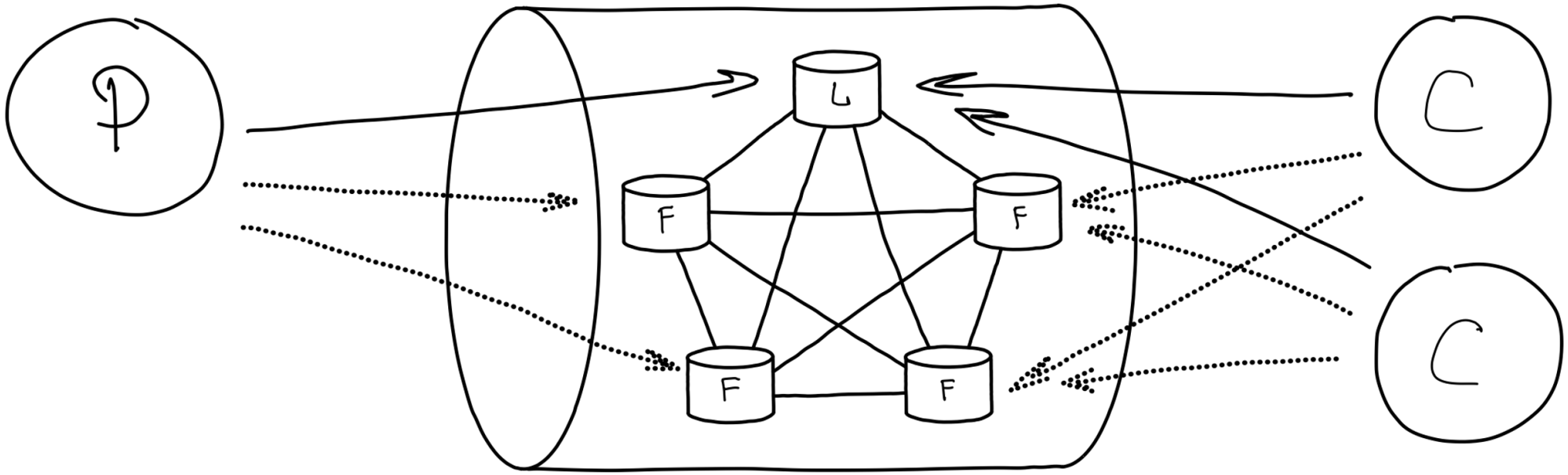
Availability: high
Durability: high

“Like a database”: quorum



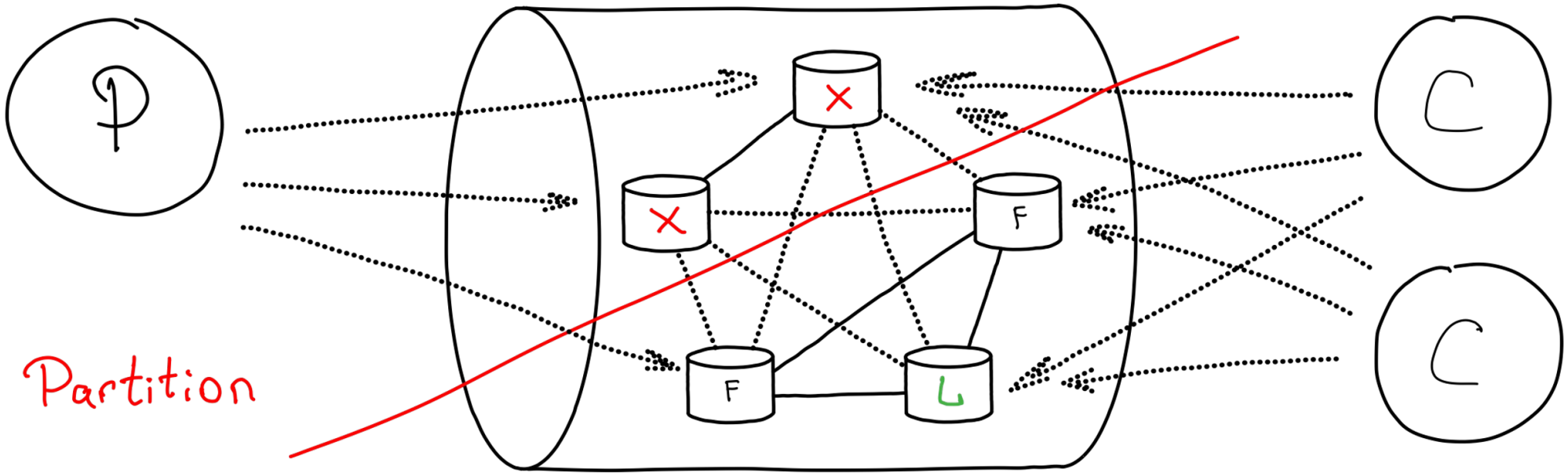
Quorum writes give protection from data loss
Quorum guarantees consistency, X → 1 (X ≥ 1)
Most reliable. But slow.

Quorum queue cluster



Guarantee: X ≈ 1 (X ≥ 1) Durability: high

Quorum queue cluster

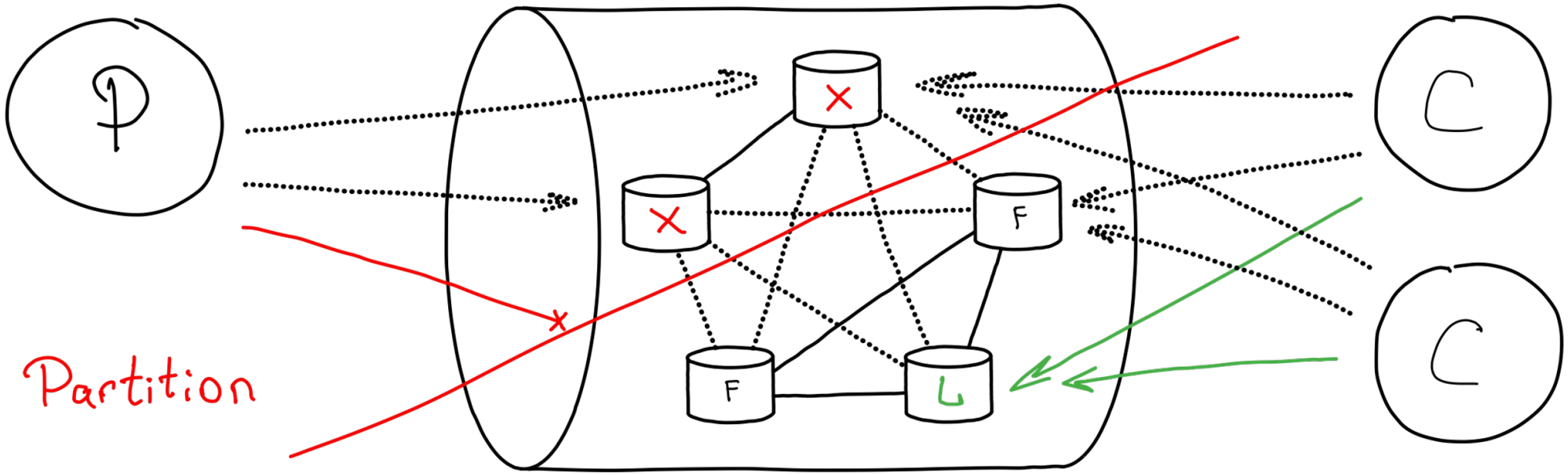


Guarantee: X ≈ 1 (X ≥ 1)

Availability: limited

Durability: high

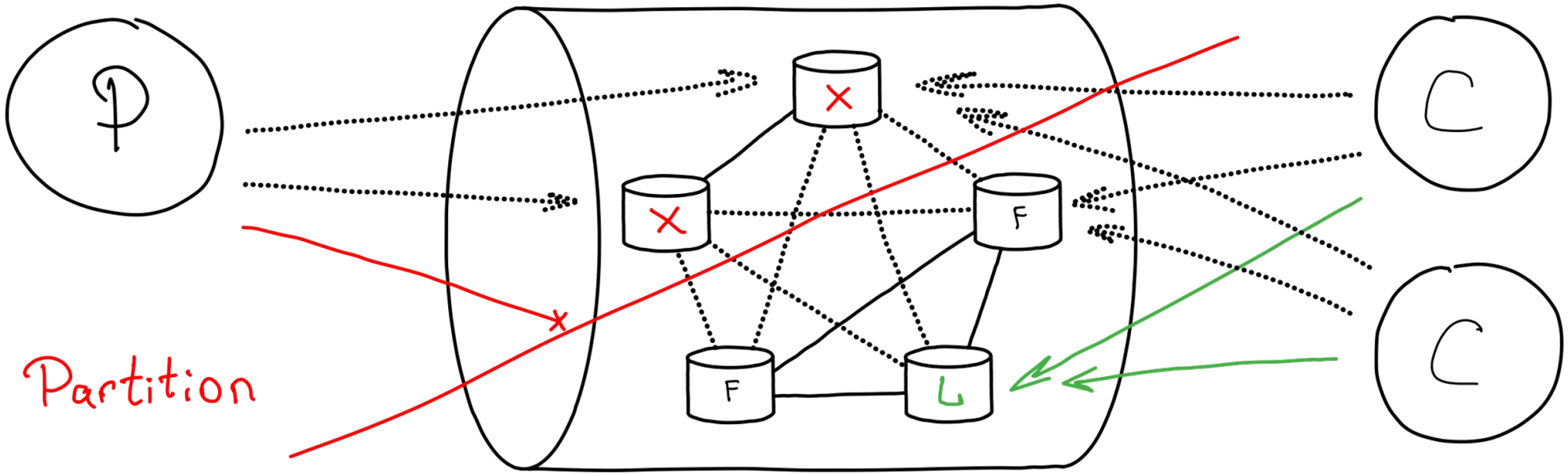
Quorum queue cluster



Guarantee: X ≈ 1 (X ≥ 1)

Availability: limited
Durability: high

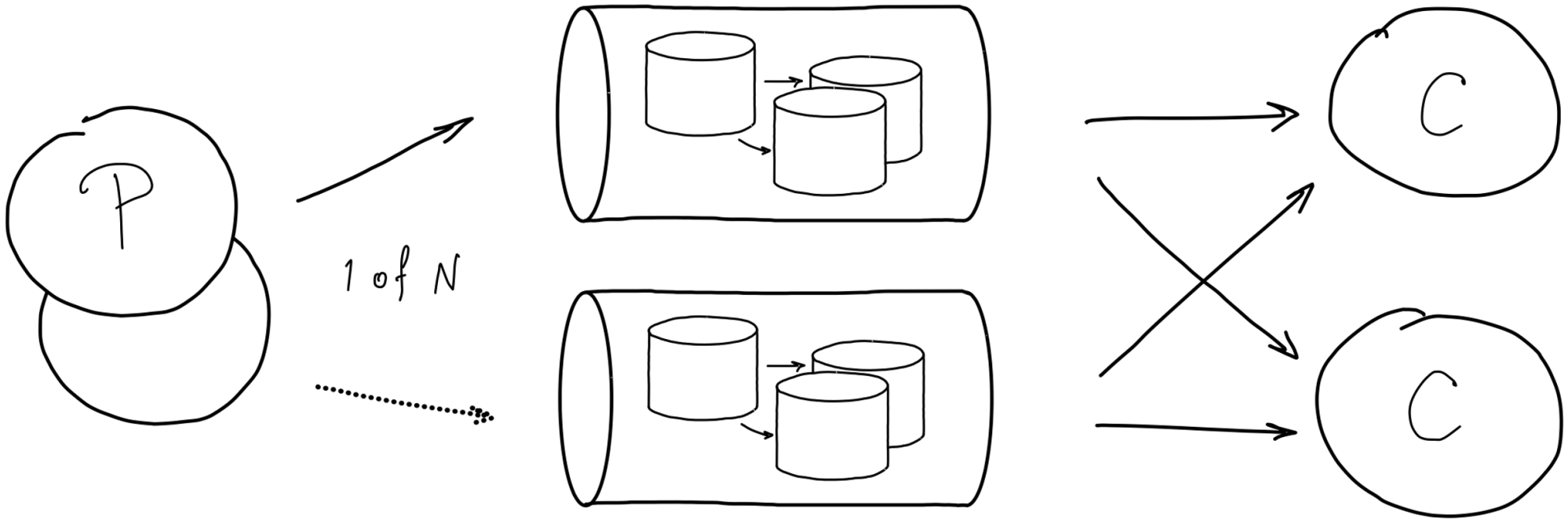
Quorum queue cluster



Scalability: no
Guarantee: X ≈ 1 (X ≥ 1)

Availability: limited
Durability: high

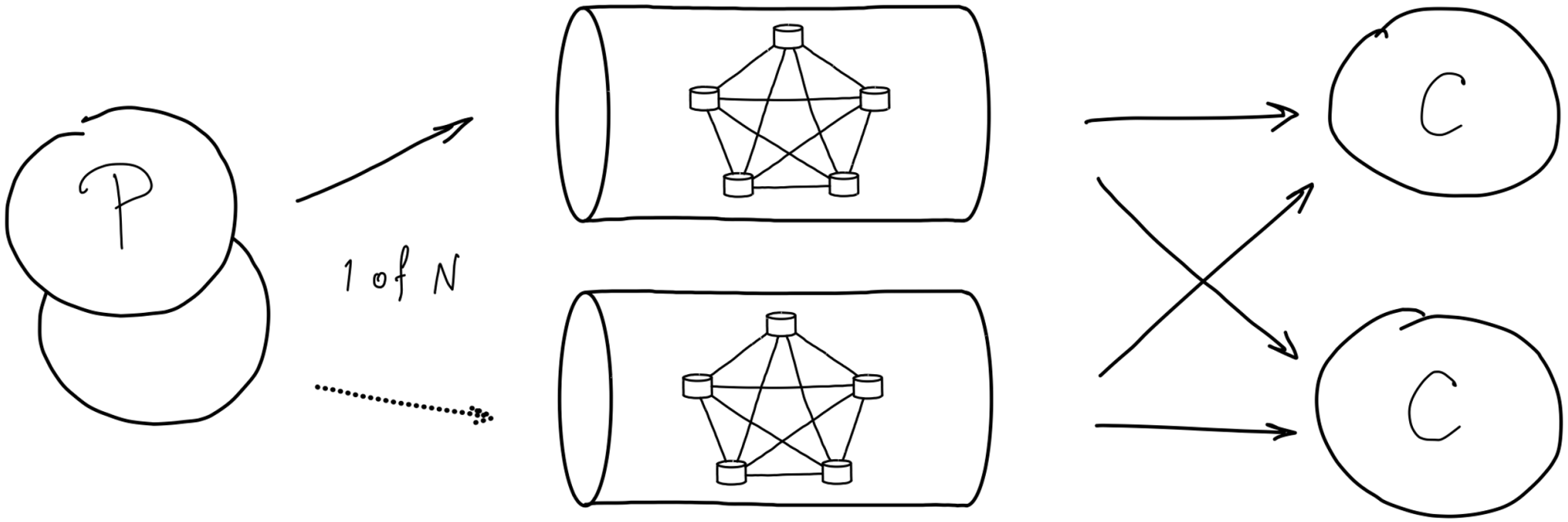
Replicated queue, 1/N



Scalability: yes
Guarantee: X ≈ 1 (X ≥ 1)

Availability: high
Durability: high

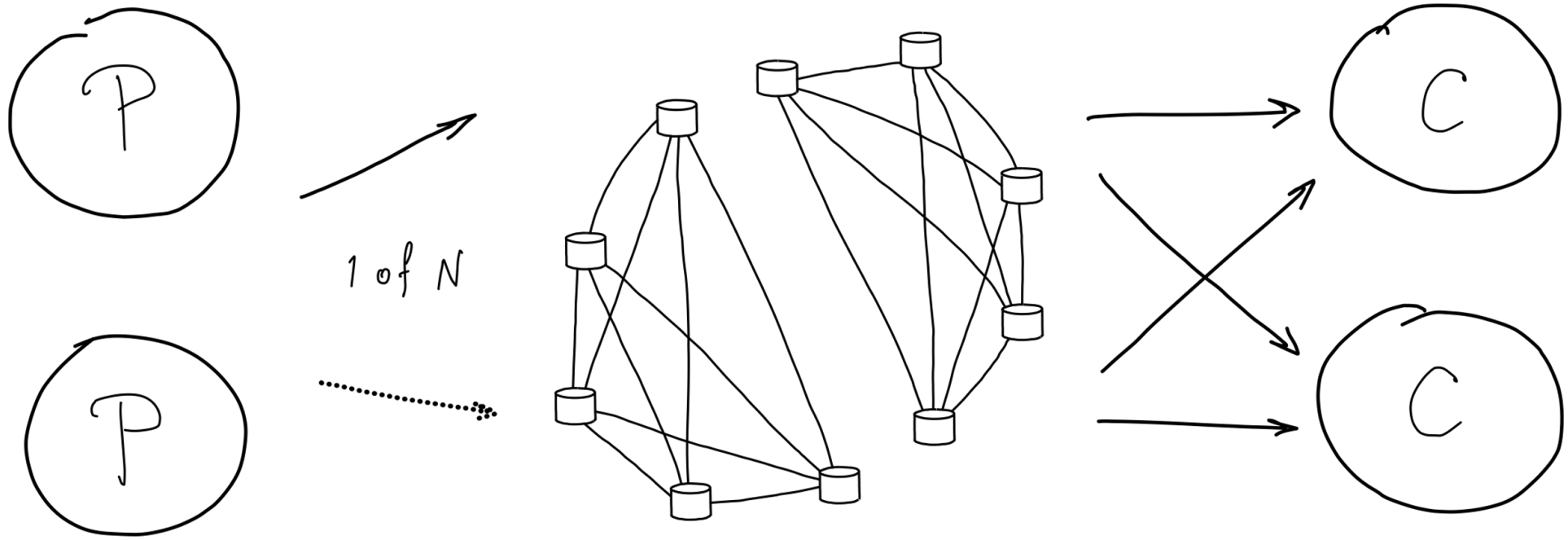
Quorum queues, 1/N



Scalability: yes
Guarantee: X ≈ 1 (X ≥ 1)

Availability: high
Durability: high

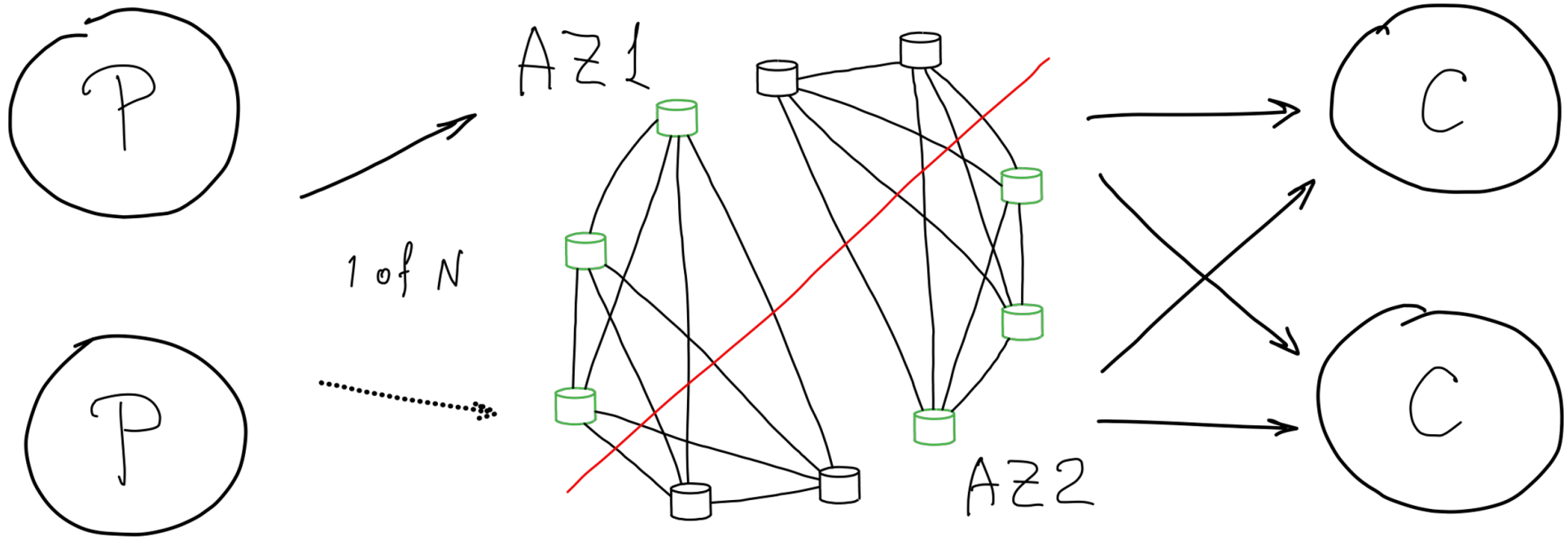
Quorum queues, 1/N



Scalability: yes
Guarantee: X ≈ 1 (X ≥ 1)

Availability: high
Durability: high

Quorum queues, 1/N



Scalability: yes
Guarantee: X ≈ 1 (X ≥ 1)

Availability: high
Durability: high

What else to know?

- Protocols & limitations
- Monitoring & maintenance
- Best choices

Protocols & limitations

Message state
is bound to connection

- Low latency
- Immediate requeue

- Hard to scale
- Task lifecycle

Stateless
(HTTP/REST/SQS)

- Scaling
- HTTP-balancing

- Autorelease is a must

Monitoring & maintenance

- Queue capacity and consumption
  - There is always a limit.

Monitoring & maintenance

- Queue capacity and consumption
  - There is always a limit.
- Timing
  - Full event cycle (QoS)
  - Consumer execution time

Monitoring & maintenance

- Queue capacity and consumption
  - There is always a limit.
- Timing
  - Full event cycle (QoS)
  - Consumer execution time
- Quantity of retries, losses and failures

Monitoring & maintenance

- Queue capacity and consumption
  - There is always a limit.
- Timing
  - Full event cycle (QoS)
  - Consumer execution time
- Quantity of retries, losses and failures
- Message flow

Monitoring & maintenance

- Queue capacity and consumption
  - There is always a limit.
- Timing
  - Full event cycle (QoS)
  - Consumer execution time
- Quantity of retries, losses and failures
- Message flow

- Duplicate your messages to logs!

Maintenance: plan a failure

- Set up failure policies

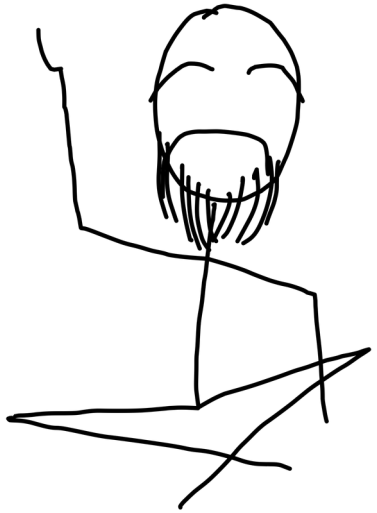
Maintenance: plan a failure

- Set up failure policies
  - Stop accepting new messages
  - Discard/drop old messages
  - Save the survivors

Maintenance: plan a failure

- Set up failure policies
  - Stop accepting new messages
  - Discard/drop old messages
  - Save the survivors

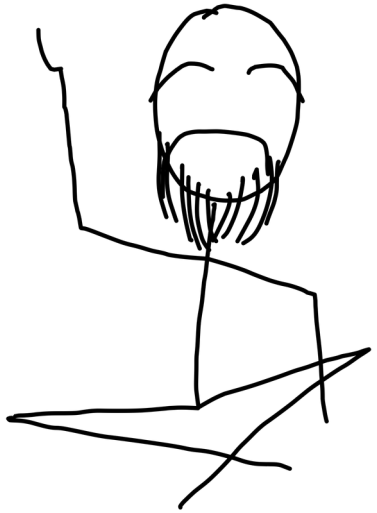
Our greatest glory is
not in never falling.
But in rising
every time we fall.



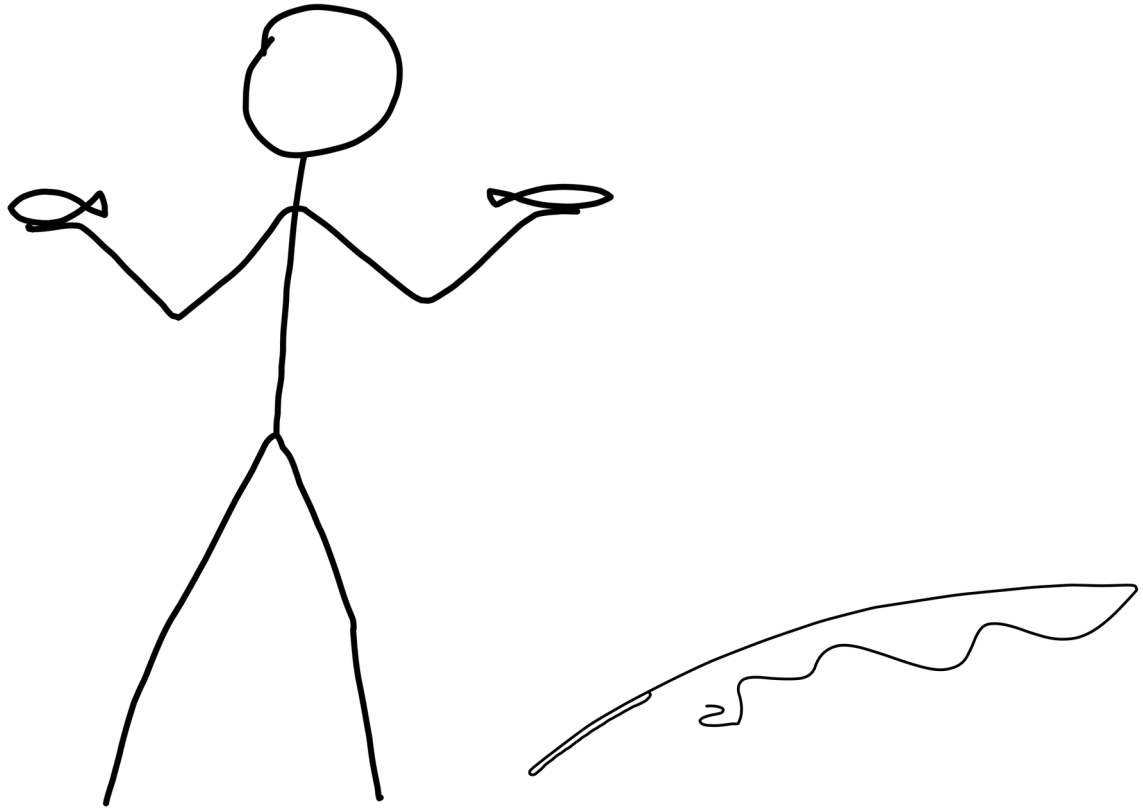
Maintenance: plan a failure

- Set up failure policies
  - Stop accepting new messages
  - Discard/drop old messages
  - Save the survivors
- Have a plan for a fall down
  - To use it to get up

Our greatest glory is
not in never falling.
But in rising
every time we fall.



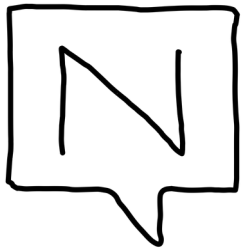
What if & how to?



What if & how to?

- Event loss tolerance
- Simple message delivery
- High throughput, scalability

What if & how to?



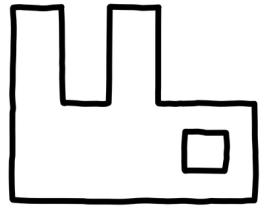
- Event loss tolerance
- Simple message delivery
- High throughput, scalability

- NATS
- NSQ
- ZeroMQ

What if & how to?

- To give it a try

What if & how to?



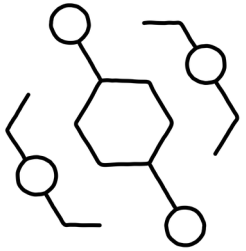
- To give it a try

- SQS (Simple Queue Service)
  - Amazon, GCP, ...
- CloudAMQP
- Simple brokers: RabbitMQ, NATS
  (watch out for durability and availability)

What if & how to?

- For cloud applications
- For microservices
- For serverless architecture

What if & how to?



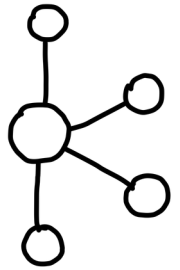
- For cloud applications
- For microservices
- For serverless architecture

- SQS (Simple Queue Service)
  - Amazon, GCP, ...
- Amazon EventBridge

What if & how to?

- Set up a streaming architecture
- High requirements for durability
- Strict FIFO

What if & how to?



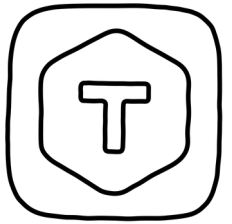
- Set up a streaming architecture
- High requirements for durability
- Strict FIFO

- Apache Kafka
- NATS JetStream
- Tarantool

What if & how to?

- Complex processing scenarios and chains
- Delayed processing, rescheduling
- Arbitrary topologies
- Dependent events

What if & how to?



- Complex processing scenarios and chains
- Delayed processing, rescheduling
- Arbitrary topologies
- Dependent events

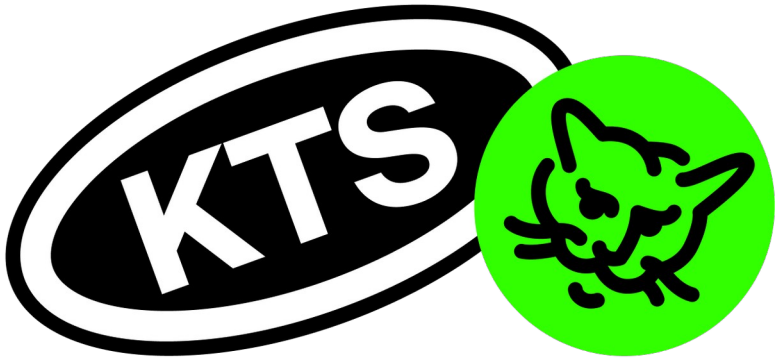
- RabbitMQ
- Tarantool Queue / Tarantool

Invitation to KTS Stand

Please join to our tech talk about queues and Tarantool

Today, 14:20

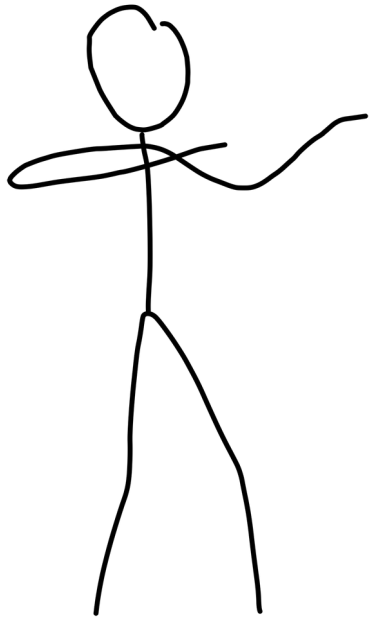
Discussion zone



Please, leave your feedback!

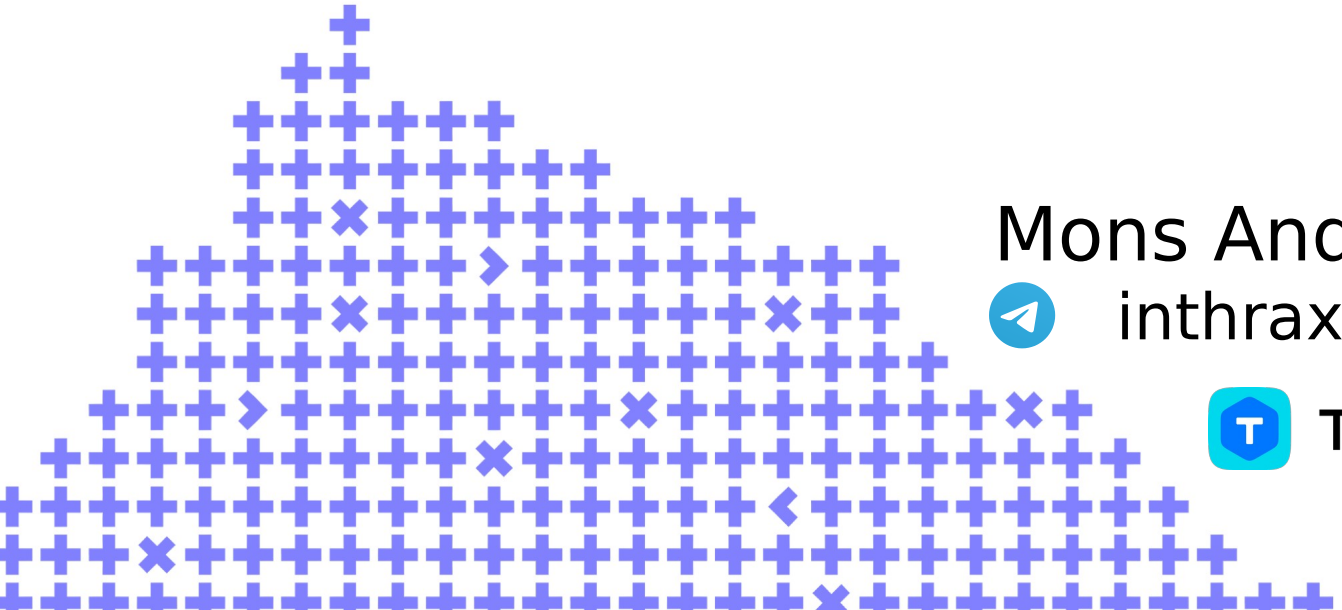
Slides:

https://bx.vc/34e3l2



Feedback:

https://bx.vc/lkmguh



Mons Anderson

inthrax

Tarantool

High Load ++ Armenia

Co-organizer

Yandex